AN ASSESSMENT OF SOFTWARE SAFETY AS APPLIED TO THE DEPARTMENT OF DEFENSE SOFTWARE DEVELOPMENT PROCESS

THESIS

Peter W. Colan, Captain, USAF
Robert W. Prouhet, Captain, USAF

AFIT/GSS/ENG/92D-2

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GSS/ENG/92D-2

AN ASSESSMENT OF SOFTWARE SAFETY AS APPLIED TO THE DEPARTMENT OF DEFENSE SOFTWARE DEVELOPMENT PROCESS

THESIS

Presented to the Faculty of the School of Systems and Logistics
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Software Systems Management

Peter W. Colan, B.S.E.
Captain, USAF

Robert W. Prouhet, B.S.
Captain, USAF

December 1992

Preface

Much has been written about the need for software safety; however, very little of the literature addresses the practical application of a software safety program. This report examines the software safety framework for DOD weapon system development and assesses the current state of software safety programs. The report is designed for system safety engineers and managers, program managers, as well as system and software engineers. While a background in DOD weapon system acquisition and software development will help the reader understand the terminology, Appendix A provides a glossary of terms.

We are indebted to a number of people for help with this work. First, our sincere appreciation goes to our thesis advisors, Dr. (Maj) Paul Bailor and Dr. Freda Stohrer, for their advice and guidance. We are particularly grateful to Mr. Mitchell Lustig, Aeronautical Systems Center (ASC) Director of System Safety, and Captain Steven Mattern, Chief of System Safety for the National Aerospace Plane. They enthusiastically provided many hours of their time for interviews and reviewed numerous drafts of our report. We also wish to thank all of the ASC System Safety Managers who granted us personal interviews and all of the telephone survey participants for their candid and detailed discussions.

Last, but certainly not least, we owe a special debt to our families. We wish to thank our wives, Carol Colan and Lynne Prouhet, for their love and support through many late nights and lost weekends. We also wish to thank our children: Megann, Brandon, Nathan, and Rachael, for their love, patience, and sacrifice of lost time with their Dads.

Peter W. Colan Robert W. Prouhet

Table of Contents

List of Figures

List of Tables

Abstract

This research analyzed the relationships between the DOD software development process, system safety requirements, and current structured software safety analysis techniques. The current state of software safety was assessed within the aerospace industry and DOD, and a training program for DOD System Safety Managers was developed.

A telephone survey was conducted to gather information on current software safety analysis techniques and methodologies. Personal interviews were conducted with Aeronautical System Center System Safety Managers to gather data on job perception and perceived training needs.

The results of the study indicate that the DOD guidance and policy documents needed to implement and manage a software safety program are adequate. Software safety programs, however, are not implemented and managed effectively; in addition, structured software safety analysis techniques are not widely used.

To improve the DOD's ability to manage weapon system safety programs, DOD should implement the proposed training program.

AN ASSESSMENT OF SOFTWARE SAFETY AS APPLIED TO THE DEPARTMENT OF DEFENSE SOFTWARE DEVELOPMENT PROCESS

I. Introduction

General Issue

Software safety is an important issue throughout the Department of Defense (DOD) in the acquisition of high technology weapon systems. As weapon system procurement approaches the twenty-first century, weapon systems are becoming increasingly expensive, complex, and computer dependent. As evidenced in the Persian Gulf War, the United States military and its allies rely heavily on computers to control critical decision making processes in weapon systems. Increasing dependence on computers to control safety-critical functions is also apparent throughout industry in areas such as air traffic control, mass transit, ship navigation, and nuclear power plant operation.

Many documented mishaps directly resulting from computer input or control problems can be found in the extensive compilation of computer related mishaps listed in references (37:6-7) and (38:xi-xiii). DOD weapon system

development has also resulted in a number of computer related mishaps. For example:

- A potentially life threatening problem with the Air Force F-16 fighter aircraft was discovered during simulation testing. When flying inverted, F-16 flight software would deadlock over the choice of a left roll or right roll to return the aircraft to level flight (34:2). Also, an F-16 was damaged during early flight testing because the computer allowed the landing gear to be raised while the aircraft was still on the ground. (35:6)

- A Navy F-18 fighter and its pilot were almost lost because a wing mounted missile failed to separate from the wing after ignition. The computer opened the missile holding clamp, fired the missile, and then closed the clamp before the missile had developed enough thrust to leave the wing. As the missile continued to thrust, the aircraft lost 20,000 feet of altitude before the pilot was able to regain control of the aircraft. (36:6)

- The Air Force's only flying prototype of the F-22 Advanced Tactical Fighter crashed due to a speculated software problem. It is hypothesized that the flight control computer was unable to move aircraft control surfaces fast enough to keep up with the pilot's commands. As of this writing, the mishap is still under investigation. (20:A3)

- The Navy has identified software problems with the F-14D aircraft development effort. A number of software defects caused cockpit displays to go blank and erroneous data to be supplied to the mission computer. (43:10)

A system failure due to software errors or inadequacies can be disastrous. "When computers are used to control safety-critical processes, there is a need to verify that the software will not cause or contribute to an accident" (4:377). Ideally, software would be tested for every conceivable input and output condition before being released for military or public use. Because of the complexity of

many safety-critical systems, however, testing for every possible input and output condition is not technically or economically feasible (44:61). In 1985 the DOD spent an estimated $11.4 billion on software development and maintenance (4:2-4). In 1990 the estimated cost for software development and maintenance was $30 billion or 10 percent of the national defense budget (22:65). The shrinking defense budget cannot continue to support the increasing cost of software development when most of the money spent is for the correction of errors (41:48).

Software Safety

Software safety is that inherent part of system safety specifically concerned with the potential safety risks associated with the software portion of a system. The relatively new concept of software safety, however, has become a concern only as a result of an increasing dependency on computers to perform functions with potentially critical safety impacts (39:636). Software safety must be evaluated "within the context of system safety" (25:223). Software is not unsafe when considered in isolation and is only an indirect contributor to mishaps (25:223). In a system context, however, the likelihood or risk that software may cause or contribute to a mishap becomes a safety-critical issue. The goal of software safety is to "ensure that software executes within the

system context without resulting in unacceptable risk" (26:35).

System safety is a discipline that seeks to identify, evaluate, and control hazards in a total system context. A hazard can be defined as "a set of conditions (i.e., a state) that can lead to an accident given certain environmental [operating] conditions" (25:223) or more simply "a condition that is prerequisite to a mishap" (8:2). Mishap is then defined as "an unplanned event or series of events that leads to an unacceptable loss such as death, injury, illness, damage to or loss of equipment or property" (8:2). Some safety engineers go even further and include environmental harm as an unacceptable loss (26:35).

The degree of risk associated with the operation of a system is classified in terms of both severity and probability of occurrence. Severity is a function of the worst possible loss associated with a hazard and probability of occurrence is a function of the likelihood that a hazard will occur and lead to a mishap (25:223). Therefore, a software system is considered safety-critical if there is some degree of risk associated with the incorrect operation of that system. Loss of human life and severe equipment damage are two examples of unacceptable risks.

Software safety should not be confused with software reliability or software security. While safety does have a relationship to both of these disciplines, there are some

distinct differences. Reliability requirements for a system are intended to ensure the system operates within a predefined failure rate. Safety requirements, on the other hand, are intended to limit the system's failures to only those failures determined to have acceptable risks. "If the design is unsafe to start with, no degree of reliability is going to make it safe" (1:20). Security requirements for a system are intended to prevent system failures due to unauthorized access and actions. Safety requirements, on the other hand, are intended to prevent system failures due to inadvertent actions (21:12).

Software Safety Requirements and Analysis

A strong emphasis must be placed on developing software requirements that include software safety. It is a well established fact in the field of software development that the vast majority of errors, including safety-critical errors, found in software systems can be directly attributed to some misunderstanding or oversight in requirements definition and design activities (40:16, 2:1). Therefore, the roots of software safety must form early in the requirements analysis phase of the software development process and maintain a significant level of managerial and technical support throughout the life of the system.

Software safety analysis is the principal means of evaluating system specific software requirements. In addition, software safety analysis can provide program

managers some level of confidence that their software will operate at an acceptable level of risk. A number of structured safety analysis techniques have been specifically developed or adapted for performing software safety analyses. These techniques, or methods, can be applied with some rigor to produce consistent analysis data. Structured software safety analysis techniques must be applied to the software that will control all weapon systems currently under development and in use to reduce the risk that software will cause a system failure which may lead to loss of life or damage to equipment.

Software Development and Safety Program Guidance

Software development policy within the DOD begins with DOD Instruction 5000.2, Defense Acquisition Management Policies and Procedures. This document specifies that software development will be governed primarily by DOD-STD-2167A, Defense System Software Development. DOD Instruction 5000.2 also specifies MIL-STD-882B, System Safety Program Requirements, as the governing document for system safety in defense acquisition programs. There are also numerous guidelines, handbooks, instructions, and regulations that provide additional insight into DOD software development and system safety. Appendix B provides a list of software development and system safety guidance and policy documents.

The documents discussed above outline what safety analyses should be performed at specific points in a software development effort and provide details on what should be covered in each of the analyses. None of the available guidance, however, adequately addresses the tools, techniques, and methodologies required to have a comprehensive software safety program. Comprehensive guidance and a firm understanding of software safety tools, techniques, and management methodologies are necessary for the implementation and management of an effective software safety program. A program manager today has no way of realistically and accurately assessing the risks that software contributes to the overall safety of a system. As systems grow more complex and computer dependent, software safety cannot be ignored.

Specific Problem

The DOD needs research to establish a comprehensive management methodology for implementing software safety programs. The components of a software safety program are the DOD software development process, DOD system safety program requirements, and current software safety analysis techniques. This thesis discusses the interrelationships of the three components of a software safety program and determines the current state of software safety analysis in weapon system development.

Research Objectives

To carry out the research defined in the last section, we divided the effort into five objectives:

1. To determine where within the DOD software development process system and software safety requirements are addressed by analyzing the software development process as outlined in DOD-STD-2167A, Defense System Software Development.

2. To determine what the current DOD system and software safety requirements are and how or if they can be easily integrated into the software development process through an analysis of MIL-STD-882B, System Safety Program Requirements.

3. To determine how or if the above software safety requirements can be met through an investigation of current software safety analysis techniques.

4. To determine what software safety analysis techniques are actually used in DOD weapon systems development and what successes or failures have been realized by their application.

5. Based on data collected in meeting objective 4, the final objective of this research was to examine the DOD System Safety Manager's (SSM) job and develop a training program to meet the SSM's hardware and software safety management needs.

II. Methodology

Introduction

This chapter describes the research methodology used to address the specific problem and objectives stated in Chapter I. The first section will describe the research design followed by a discussion of the data collection process, the population from which the data was collected, and finally, the data analysis process.

Research Design

This research was carried out in two sequential phases. The first phase consisted of two steps and was designed to lay the foundation for the study by determining how existing guidance documents and software safety analysis techniques may be combined to develop a software safety program. Step 1 involved answering the following three questions:

1. Where within the DOD software development process are system and software safety requirements addressed?
2. What are the current DOD system and software safety requirements?
3. How can the software safety requirements be met?

These questions correspond to research objectives 1, 2, and 3 stated in Chapter I.

Step 2 involved a survey of aerospace industry and DOD personnel to assess the current state of software safety within DOD weapon system development and determine what software safety analysis techniques are actually being used.

This step corresponds to research objective 4 stated in Chapter I.

The second phase of this research explored training and education requirements for System Safety Managers located at Aeronautical Systems Center, Wright-Patterson AFB, Ohio. This phase corresponds to research objective 5 stated in Chapter I.

Data Collection

The data collection for this research involved two phases. Table 1 shows a matrix of research objectives and data collection methods. In phase one, we gathered data to address the first four research objectives. The first three objectives were fulfilled primarily by literature review; the results are reported in Chapter III. The fourth objective was fulfilled by conducting a telephone survey; the results are reported in Chapter IV.

Table 1. Research Objective Methodology Matrix

Phase	Research Objective	Literature Review	Telephone Survey	Document Review	Personal Interview
1	1	X			
	2	X			
	3	X	X		
	4		X		
2	5			X	X

The second phase of data collection specifically addressed research objective 5. Data collection for this objective consisted of reviewing system safety training material, system safety guidance and policy documents, and weapon system hazard analysis reports. In addition, SSMs were interviewed for their perceptions of the system safety manger's job. Our analysis of SSM training and education is presented in Chapter V.

Literature Review. DOD guidance and requirements documents served as the principal means of describing and analyzing the DOD software development process and the system safety program requirements. The open literature provided the principal means of identifying and describing current software safety analysis techniques.

Telephone Survey. The telephone interview was selected as our survey methodology. Because the application of structured safety analysis techniques to software is a relatively new practice, determining the extent and effectiveness of their use required screening a large pool of sources within the aerospace industry and DOD. We needed to determine quickly the true subject knowledge held by potential participants. C. William Emory, in Business Research Methods, suggests that telephone surveys tend to be the quickest and most economic method to reach and screen a large pool of potential participants (15:318). While the telephone interview does limit the length of interviews and

the complexity of questions, it has the additional advantage of reducing interviewer bias (15:331-332).

Interviews were conducted as semi-structured, focused interviews. We developed a set of questions designed to guide the topical direction and coverage of the interview. Focus questions were developed to facilitate identification of subject area experts, to gain a perspective on the state of software safety analysis within DOD software development, and to keep the discussions on track. Additional questions were included for administrative and participant characterization purposes. The survey questionnaire is included in Appendix D.

The unstructured nature of the interviews established an open and flexible atmosphere and allowed us to probe as deeply as necessary to gain a full understanding of the topic. Survey participants were briefed at the start of each interview about the objectives of the research. With each question, participants were allowed to respond in an unstructured, open-ended format. Specific follow up questions were composed as the interviews progressed.

Population

The target population for this study consisted of all aerospace industry and DOD representatives that could be classified as one or more of the following:

- Recognized experts in software safety analysis
- System/software safety engineers and/or managers

- Project engineers and/or managers

No attempt was made to determine the size of this population, but it probably consists of thousands of individuals. Also, no upper limit was set on the number of individuals that would be included in this study.

Our goal was to contact as many individuals as possible in order to cover a fairly broad spectrum within the aerospace industry and DOD. Fifty-four individuals within the target population were contacted. However, only 23 were experienced in software development and/or software safety. These individuals became the sample population. A list of the individuals included in the sample population is provided in Appendix E.

Data Analysis

Data obtained from the telephone interviews were used to describe the current state of software safety analysis as part of DOD software development efforts. The data were analyzed for three kinds of information: First, is software safety important to system and software development efforts? Second, what software safety analysis techniques are employed and what success or failure has been realized? Third, what benefits and/or drawbacks are associated with software safety analysis techniques?

To determine the overall importance of software safety to a software or system development effort, survey participants provided subjective assessments of the

importance of software safety in their organizations. An important part of these assessments was whether or not software safety was an integral part of each organization's development process. This information helped determine the general level of industry and DOD awareness of software safety.

Survey participants were asked what software safety analysis techniques were used by their organizations. Responses determined which techniques were actually being used and identified other techniques not encountered during initial literature review. Many analysis techniques were identified, however, only three had been developed or adapted specifically for software safety analysis. Further literature review provided descriptions of these newly identified structured analysis techniques.

Survey participants were also asked to recount any successes or failures realized by the application of structured software safety analysis techniques. Success was defined as the application of an analysis technique that resulted in a documented statistical reduction in system risk. Participants were asked to provide their own definition of failure. Survey responses were assessed to determine the use and viability of structured software safety analysis techniques.

Survey participants were asked to assess the benefits or drawbacks associated with the application of software

safety analysis techniques. These data were used to identify strengths and weaknesses of the analysis techniques, potential problems in the management of software safety programs, and potential training deficiencies encountered by safety engineers and managers.

III. Literature Review

Introduction

This literature review analyzes the DOD software development process and DOD system safety program and investigates current software safety analysis techniques. Figure 1 shows the three components of a software safety program. Our analysis of the literature will determine where system and software safety requirements are addressed within the software development process, what the system and software safety program requirements are, and how the software safety analysis requirements are met.

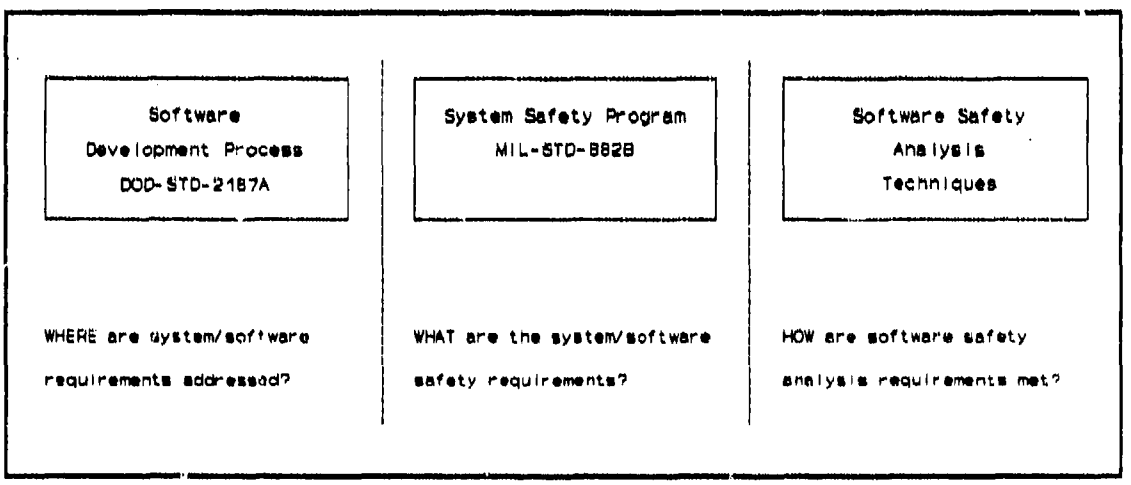

Figure 1. Software Safety Program Components

Although software safety applies to all computer controlled systems, this literature review focuses only on software safety as it applies to the acquisition of weapon systems within the DOD.

This chapter is divided into three sections. The first section analyzes the DOD software development process as outlined in DOD-STD-2167A, Defense System Software Development. Section two analyzes the DOD system safety program as outlined in MIL-STD-882B, System Safety Program Requirements. Each task section within the standard is discussed as it relates to the implementation and management of a system safety program and the software development process. Section three describes current software safety analysis techniques.

DOD Software Development Process

A process in its simplest terms is a "set of operations occurring in a definite sequence that operates on a given input and converts it to some desired output" (16:299). Taken a step further, a software development process may be defined as a set of "software engineering activities, including technical and managerial ones, that are carried out in the production of software" (31:234).

There are several software development models and methods available to guide a software development project. This research focuses on only one: "the classic life cycle paradigm for software engineering," this model "is the oldest and the most widely used paradigm for software engineering" (40:20-21). This paradigm, commonly referred to as the "Waterfall Model" takes a "sequential approach to software development that begins at the system level and

progresses through analysis, design, coding, testing, and maintenance" (40:20). The waterfall model has been adopted by the DOD and used to guide the software development process for many major weapon systems.

Every software development project presents a unique set of circumstances and system requirements which determine the amount of detail required in each technical and managerial phase of its development. "The goal of the software development process" therefore, "is to consistently produce a quality product . . ." (5:5-1). DOD-STD-2167A is the primary guide for DOD software development projects and has expanded the traditional waterfall model into eight distinct development phases.

The eight software development phases are shown in Figure 2. Although the process must stay relatively constant, the development phases often occur concurrently (5:5-2). The first phase, System Requirements Analysis/Design and the last phase, System Integration and Testing are common to hardware and software. There are six intermediate phases peculiar to software. For this research all eight phases are discussed as they relate to software development and safety. During the software development process, development activities are monitored through a series of formal reviews and audits as described in MIL-STD-1521B, Technical Reviews and Audits for Systems, Equipment, and Computer Systems. Figure 2 shows how the

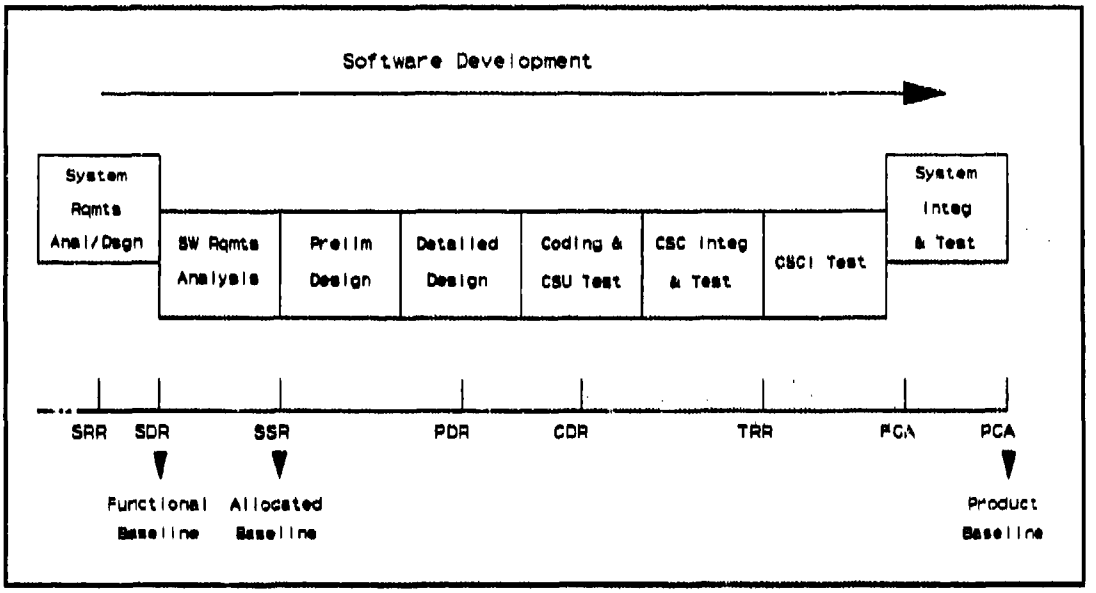

Figure 2. Software Development Process (5:5-2)

formal reviews and audits relate to the eight software development phases. In addition, the figure shows the three system baselines used to maintain configuration control of the system under development. The acronyms for each review and audit as well as baseline definitions are included in Appendix A.

The development process for a weapon system begins by determining system requirements. Concept Exploration and Demonstration/Validation are the first two phases of the system acquisition cycle which may be conducted to establish system level requirements if no previous requirements exist. Hardware and software development then take place during the Full Scale Development phase of the system acquisition

cycle. The software development process actually begins while the system level requirements are analyzed and an initial allocation of system functions is made to hardware and software.

System Requirements Analysis/Design. The System Requirements Analysis/Design phase refines the system level requirements and establishes the system's functional baseline (5:5-4, 17:12). Requirements refinement may be accomplished by gathering information through independent analysis, trade studies, and prototype development (17:12). The functional baseline is established by the approval of

> ... the system specification and the operational concept; by developing the initial subsystem/segment designs; and by refining the systems engineering planning activities to be employed during system's development. (5:5-4,5-5)

Expected outputs from System Requirements Analysis/Design are

- System Specifications
- System/Segment Design Documents
- Software Requirements Specification (Preliminary)
- Interface Requirements Specification (Preliminary)
- Operational Concept Document
- Software Development Plan
- Configuration Management Plan (5:5-3).

The System Requirements Analysis/Design phase concludes with a System Requirements Review (SRR) and a System Design

Review (SDR). At the SRR the developer's progress toward understanding and defining the system level requirements is evaluated (5:5-2). Two important topics discussed at this meeting specifically relating to software safety are the program risk analysis and the system safety analysis conducted by the contractor (9:19-20). These analyses can provide important information concerning the potential hazards and risks associated with the design and operation of the system under development and should be used to develop requirements to eliminate or control hazards (9:21).

At the SDR all system level requirements are reviewed in preparation for establishing the functional baseline (5:5-2). Once the functional baseline is established, the development process can move on to the next phase. An important part of the SDR is the review of failure-mode and effects analyses to ensure identification and ranking of technical and program risks (9:24). In addition, system hazard analyses are reviewed with an emphasis on identification of safety test requirements (9:25). The results of the developer's analyses become an important part of the program manager's decision to progress to the next phase of development.

Software Requirements Analysis. The Software Requirements Analysis phase establishes "detailed functional, performance, interface, and qualification

requirements for each CSCI [Computer Software Configuration Item] based on the System Specification" (5:5-6). This phase also identifies system level testing requirements (5:5-6).

Expected outputs from Software Requirements Analysis are

- Software Requirements Specification
- Interface Requirements Specification
- Software Development Plan (Updated) (5:5-6).

The Software Requirements Analysis phase concludes with a Software Specification Review (SSR). The SSR covers each CSCI's software requirements as outlined in the Software Requirements Specification and the Interface Requirements Specification (9:31). This review also provides another opportunity for risk assessment in regards to software safety by examining updates to previously delivered safety data (9:31-32). Successful completion of the SSR establishes the allocated baseline.

Preliminary Design. The Preliminary Design transforms system and software requirements into some type of software architecture (40:215). During this phase the contractor uses the outputs from the previous development phases and "partition[s] the software into components and define[s] the function of each component and the relationships between them" (5:5-7).

Expected outputs from Preliminary Design are

- Software Design Document (Preliminary)
- Software Test Plan (Test Identification)
- Interface Design Document (Preliminary) (5:5-7).

The Preliminary Design phase concludes with a Preliminary Design Review (PDR). The PDR is a comprehensive review of the entire system and provides the first in-depth review of system safety issues. Specific safety items to be reviewed are

- results of qualitative and quantitative hazard analyses;
- results of system and intra-system safety interface trade-off studies;
- safety requirements passed down to subcontractors;
- special or system peculiar safety areas;
- adequacy of preliminary design for meeting safety requirements (9:42).

The PDR assesses the feasibility of the developer's design, assesses current risks, and determines whether the development effort is ready to progress into the next phase (9:33).

Detailed Design. "The purpose of the Detailed Design activity is to logically define and complete a software design that satisfies the allocated requirements" (5:5-8). During this phase, the developer refines the preliminary design to a level of detail that allows development of the computer program by someone other than the designer (5:5-8).

If there are any deviations from the software requirements specification, they should be defined, justified, and submitted to the government for approval.

Expected outputs from Detailed Design are

- Software Design Document (Detailed Design)
- Software Test Description (Cases)
- Interface Design Document (5:5-8).

The Detailed Design phase concludes with a Critical Design Review (CDR). The CDR "assure[s] that the software design satisfies the requirements of both the system level specification and the software development specification" (5:5-9). System safety items of particular concern at the CDR include

- review of detailed design for safety design requirement compliance;
- review acceptance test requirements for inclusion of adequate safety requirements;
- review operational maintenance safety; analyses and procedures (9:61).

Successful completion of CDR enables the developer to begin the next phase.

Coding and Computer Software Unit Testing. Coding translates the detailed software design into a specific programming language (5:5-9). Source code listings are generated as the program is completed. It is the programmer's responsibility to examine completed Computer Software Unit (CSU) code for errors (5:5-9). If considered

error free and capable of meeting requirements, the code is compiled and an object code listing is obtained.

After coding, but before compilation, each CSU must be tested to eliminate any errors that may exist (5:5-9). Particular attention is paid to the algorithm and logic implemented as well as to the CSU's ability to satisfy requirements (7:27). Specific informal test procedures are generated and recorded in the CSU's Software Development Folder (SDF) along with the results of testing. If errors are found, the software requirements and design documents might be revised and the CSU is re-coded and re-tested. Finally, test procedures are developed for testing the Computer Software Component (CSC) that a particular CSU is part of (7:27). These CSC test procedures are recorded in the CSC's development folder.

Expected outputs from CSU Coding and Testing are

- CSU source and object code listings
- CSU test procedures and results
- CSC test procedures (7:27).

No formal review is associated with the Code and CSU Testing phase. CSU coding and testing leads directly to Computer Software Component Integration and Testing.

Computer Software Component Integration and Testing. Computer Software Component Integration and Testing is conducted to "combine the software units and components that have been independently tested into the total software

product and to demonstrate that this combination fulfills the system design" (5:5-10). CSUs are integrated into CSCs which in turn may be integrated into larger CSCs. Each combination is tested to ensure that the implementation algorithm and logic are error free and that requirements are satisfied (7:29). Test results determine whether requirements and design documents need to be updated and CSUs re-coded and re-tested (7:29). In preparation for Formal Qualification Testing (FQT), each test case in the Software Test Description (STD) should be pretested during this phase (7:29). Test procedures for CSCI testing are also developed at this time.

Expected outputs from CSC Integration and Testing are

- Software Test Description (Procedures)
- CSC test results (7:29).

The CSC Integration and Testing phase concludes with a Test Readiness Review (TRR) (5:5-10). At the TRR, the results of informal testing are reviewed and software test procedures, including safety test procedures, are evaluated for adequacy and compliance with test plans (9:69). The TRR is used to ascertain whether or not the development is ready to progress to CSCI Testing (9:69).

Computer Software Configuration Item Testing. CSCI Testing is conducted to test each CSCI using the software test plans and procedures written earlier in the development. CSCI testing constitutes Formal Qualification

Testing and serves to establish the product baseline (9:31). Each CSCI requirement in the Software Requirement Specification and Interface Requirement Specification must be verified. Results of testing are recorded in Software Test Reports and any errors found are analyzed for their system and safety impacts. If deemed necessary, requirements and design documents are updated and CSUs, CSCs, and CSCIs are re-coded and re-tested (9:31).

Expected outputs from CSCI Testing are

- Software Test Reports
- [Final] Software Product Specification
- [Final] Source Code Listings
- [Final] Software Design Document (5:5-11).

By completion of CSCI Testing, the Software Product Specification (SPS) and all support documents necessary to operate and maintain the fielded software product are finalized. Support documents include

- Computer System Operator's Manual
- Software User's Manual
- Software Programmer's Manual
- Firmware Support Manual
- Computer Resources Integrated Support Document
- Version Description Document (5:5-11).

The CSCI Testing phase typically concludes with the Functional Configuration Audit (FCA) followed by the Physical Configuration Audit (PCA) (5:5-11). The FCA and

PCA may, however, be run concurrently and are sometimes deferred until after System Integration and Testing. The FCA verifies that each CSCI performs in accordance with its requirements and interface specification (9:69). The software test reports are examined and the appropriate operations and support documents are reviewed (5:5-11). The PCA validates the finished software product against its design documentation (9:75). The audit involves direct comparison and examination of the SPS and the source code listing. Successful completion of the PCA establishes the product baseline against which all subsequent changes must be made by formal engineering change proposals (9:75).

System Integration and Testing. The System Integration and Testing phase ensures that the newly developed software will work with the overall system in an operational environment (5:5-12). The hardware and software performance as well as system interfaces are examined and compared to system-level specifications (9:83). Often, the FCA and PCA are conducted during this phase.

The expected outputs of System Integration and Testing are any final updates to requirements, design, and interface documents. In addition, all documentation and source and object code listings should be delivered (7:33).

The System Integration and Testing phase ends with a Formal Qualification Review (FQR). "The FQR is a

system-level review that verifies that the actual system performance complies with system [including safety] requirements" (5:5-12).

Software Development Process Summary. DOD-STD-2167A defines a detailed, highly structured software development process. The process description emphasizes process steps and their expected outputs in the form of formal documents and software products. Program managers have the authority to tailor the process to fit the peculiar needs of their program. Although DOD-STD-2167A makes a few references to system safety, it does not specifically address software safety.

MIL-STD-1521B also defines detailed and highly structured reviews and audits, but unlike DOD-STD-2167A, MIL-STD-1521B discusses in detail those safety items that should be reviewed at specific points in the development process. Each formal review and audit indicates specific system and software safety items for review. Program managers, however, can limit the specific items to be discussed at any review or audit.

This analysis of the software development process along with its formal reviews and audits reveals significant emphasis on system and software safety. The principal limitation of the development process, from a safety standpoint, is that potentially all of the safety review can be tailored out of the process.

DOD System Safety Program

A system safety program is an essential element in the DOD's acquisition of high technology weapon systems. The primary document currently in use by the DOD to impose requirements for developing and implementing a system safety program is MIL-STD-882B with Notice 1. According to MIL-STD-882B:

> The principal objective of a system safety program within the Department of Defense (DOD) is to make sure safety, consistent with mission requirements, is designed into systems, subsystems, equipment, and facilities, and interfaces. (8:iii)

To accomplish this objective, MIL-STD-882B provides general system safety program guidance, definitions, and requirements as well as specific system safety tasks contained in three task sections: Task Section 100, Task Section 200, and Task Section 300. Each task section consists of several specific tasks which are selectively applied to a development effort. Tasks may be used as written or tailored to meet the specific needs of the program. The degree to which these tasks are levied against the system development contractor is the responsibility of the System Safety Manager (SSM) (8:3).

System Safety Program Requirements. A totally safe system is often not an affordable option in a major weapon system acquisition. Therefore, a system is usually designed to an acceptable degree of safety. This degree of safety implies some inherent risk in the system. The emphasis

within system safety is to identify and analyze potential system hazards to determine what level of risk each poses. Efforts are then made to eliminate or reduce these risks. It is the program manager's responsibility to specify the acceptable level of risk for a program.

Identified hazards are controlled systematically. The first step is to eliminate the hazard or reduce its associated risk through design. Next, if the hazard cannot be designed out, it must be controlled by designing and installing safety devices or it should be isolated via the design and installation of protective systems. The third step is to provide systems and devices that detect the hazard condition and produce a warning signal. Finally, for those hazards that can neither be eliminated nor have their associated risk reduced, special procedures and training to implement the procedures must be developed and included in technical manuals (8:6).

In order to assess the risk of any hazard, the hazard must be categorized in terms of its severity and probability of occurrence. MIL-STD-882B defines hazard severity as

> ... a qualitative measure of the worst credible mishap resulting from personnel error; environmental conditions; design inadequacies; procedural deficiencies; or system, subsystem or component failure or malfunction .... (8:6-7)

Table 2 shows some sample hazard severity categories that may be used by program managers as guidance when defining severity categories for their system.

Table 2. Hazard Severity Categories (8:7)

Category	Description	Mishap Definition
I	CATASTROPHIC	Death or system loss.
II	CRITICAL	Severe injury, severe occupational illness, or major system damage.
III	MARGINAL	Minor injury, minor occupational illness, or minor system damage.
IV	NEGLIGIBLE	Less than minor injury, occupational illness, or system damage.

Hazard probability is defined as "the probability that a hazard will be created during the planned life expectancy of the system . . ." (8:7). Potential hazard occurrences can be measured per units of time or by other events and categories of interest. Hazard probabilities are fairly easy to measure in a mature system in which testing or operation has begun. Early in a program's development, however, hazard probabilities must be obtained through a combination of research and analysis, to include the evaluation of historical data. Table 3 shows some sample hazard probability levels. Again, the program manager must determine acceptable or unacceptable hazard probability rankings.

It is important to note that the general system safety program requirements do not differentiate between hardware and software. The requirements emphasis is on a safe system. The task sections of MIL-STD-882B provide the means

Table 3. Hazard Probability Rankings (8:7)

Level	Description	Specific Individual Item	Fleet or Inventory
A	FREQUENT	Likely to occur frequently	Continuously experienced
B	PROBABLE	Will occur several times in life of an item	Will occur frequently
C	OCCASIONAL	Likely to occur sometime in life of an item	Will occur several times
D	REMOTE	Unlikely but possible to occur in life of an item	Unlikely but can reasonably be expected to occur
E	IMPROBABLE	So unlikely, it can be assumed occurrence may not be experienced	Unlikely to occur, but possible

to tailor a system safety program for both the hardware and software of the system being developed.

Task Section 100. Task Section 100 provides specific program management and control tasks that may be employed to establish a system safety program for a system development effort. Table 4 lists the 100 series tasks. In addition to the task listings, Appendix A of MIL-STD-882B provides guidance to the SSM in the task selection and implementation (for 100, 200, and 300 series tasks) required to establish and manage a system safety program.

From a software safety perspective, Task 101, System Safety Program Plan, is the most important of the 100 series tasks. This task requires the developer to describe his approach to system safety. Specific procedures and analysis

Table 4. Program Management and Control Tasks

Task Number	Title
100	System Safety Program
101	System Safety Program Plan
102	Integration/Management of Associate Contractors, Subcontractors, and Architect and Engineering Firms
103	System Safety Program Reviews
104	System Safety Group/System Safety Working Group Support
105	Hazard Tracking and Risk Resolution
106	Test and Evaluation Safety
107	System Safety Progress Summary
108	Qualifications of Key Contractor System Safety Engineers/Managers

methodologies are presented to the program office for review and approval. The System Safety Program Plan provides information on the developer's conceptualization of the hardware and software relationships in a system and how software safety will be treated in relationship to system safety.

Task Section 200. Task Section 200 provides specific design and evaluation tasks. The 200 series tasks are designed to provide guidance for hardware safety analyses. However, these tasks may be used to assure some level of software safety in systems where software plays a minor role and separate software analyses are not economical (1:21). Table 5 lists the 200 series tasks.

Table 5. Design and Evaluation Tasks

Task Number	Title
201	Preliminary Hazard List
202	Preliminary Hazard Analysis
203	Subsystem Hazard Analysis
204	System Hazard Analysis
205	Operating and Support Hazard Analysis
206	Occupational Health Hazard Assessment
207	Safety Verification
208	Training
209	Safety Assessment
210	Safety Compliance Assessment
211	Safety Review of Engineering Change Proposals and Requests for Deviation/Waiver
212	Reserved
213	GFE/GFP System Safety Analysis

The initial phases of a system development involve evaluating those hazards that a system might encounter so that the remaining system safety work effort can be defined (8:A-11). Task 201, Preliminary Hazard List (PHL), and Task 202, Preliminary Hazard Analysis (PHA), call for a preliminary look at the entire system involving both hardware and software. Thorough PHLs and PHAs form the "framework for other hazard analyses which may be performed" (8:A-11). For example, the PHL and PHA form the foundation for subsequent implementation of Task 301, Software Requirements Hazard Analysis.

The PHL is merely a compilation of possible hazards taken from historical records and some brainstorming by the analysts. An initial determination of hazard significance is also made. The PHA is a continuation of the PHL. It identifies safety critical areas in the system through an initial hazard evaluation, and identifies safety design criteria. Specific system concerns for a PHA include hazardous components; safety related interfaces (includes software controls); environmental constraints; operating, test, maintenance and emergency procedures; facilities and support equipment; and safety equipment (8:201-1, 202-1).

The next three tasks are all extensions of the PHA. Task 203, Subsystem Hazard Analysis (SSHA), extends the PHA to identify hazards associated with the design of subsystems. Task 204, System Hazard Analysis (SHA) analyzes system level hazards including subsystem interfaces. Task 205, Operating and Support Hazard Analysis (O&SHA), analyzes operating and support hazards (8:203-1, 204-1, 205-1).

Another task with potential software safety implications is Task 207, Safety Verification. The purpose of this task is "to define and perform tests and demonstrations or use other verification methods on safety critical hardware, software, and procedures . . ." (8:207-1). This task allows verification of the successful elimination or control of safety hazards.

Task Section 300. The 300 series tasks "specify the types of software safety analyses that should be performed as part of a comprehensive safety program" (1:21). The 300 series tasks are listed in Table 6. These tasks, added in July 1987 and listed in Notice 1 to MIL-STD-882B, complement the software development process as outlined in DOD-STD-2167A (1:21). Figure 3 shows how these tasks correspond to the software development process. The 300 series tasks were developed to provide increased emphasis on software safety engineering functions and analysis throughout the entire software development process (32:27).

Table 6. Software Hazard Analysis Tasks

Task Number	Title
301	Software Requirements Hazard Analysis
302	Top-Level Design Hazard Analysis
303	Detailed Design Hazard Analysis
304	Code-Level Software Hazard Analysis
305	Software Safety Testing
306	Software/User Interface Testing
307	Software Change Hazard Analysis

The Software Requirements Hazard Analysis (SRHA), Task 301, is the first and perhaps the most important task specifically designed to insure some level of software safety within a system. Its purpose "is to develop safety design requirements to be included in the system and

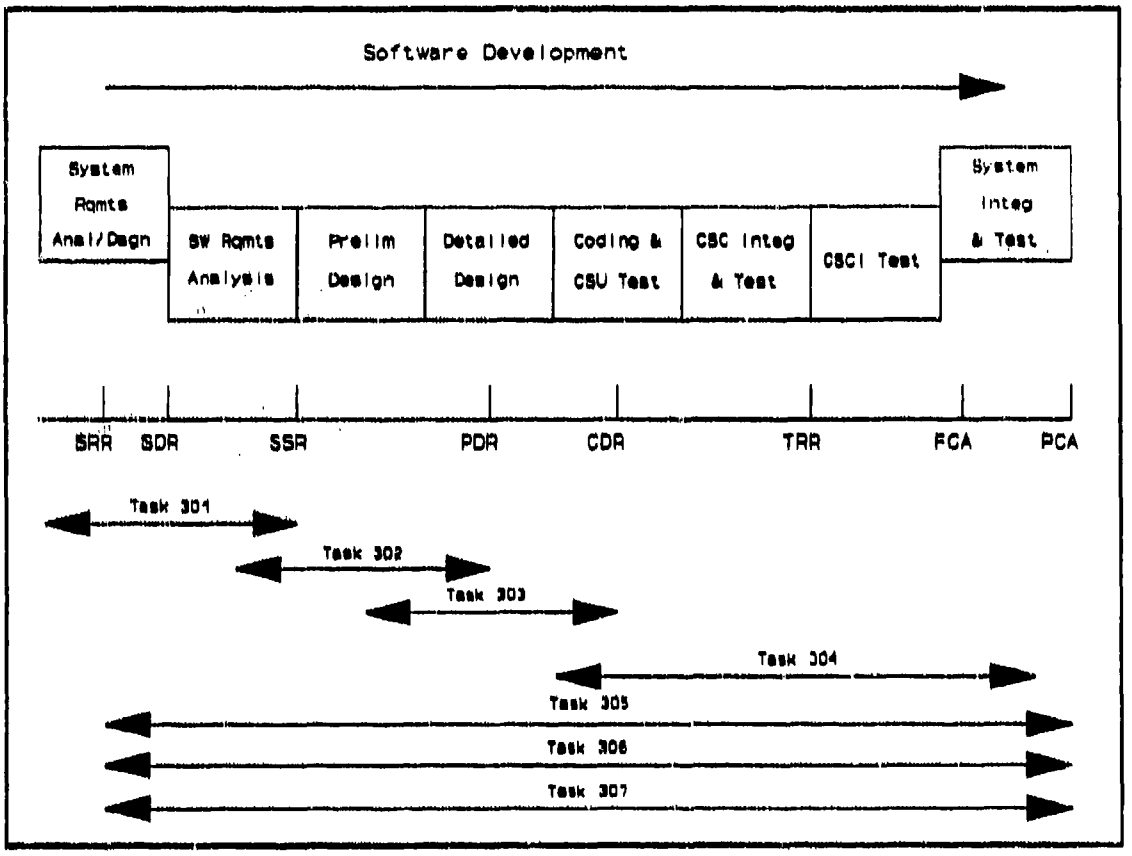

Figure 3. Relationship between 300 Series Tasks and the Software Development Process (1:22)

software preliminary design" (1:22). To meet the preliminary design deadline, the SRHA system safety requirement must be levied against the developer early in the System Requirements Analysis/Design phase of the development process.

Basic inputs into the SRHA are the risks identified through the Preliminary Hazard List (Task 201), and the Preliminary Hazard Analysis (Task 202) (8:301-1). Outputs from the SRHA are used to build the foundation for system

level safety requirements and serve as inputs to the succeeding 300 series safety analyses (1:22, 8:301-1).

Tasks 302 through 306 parallel different phases of the development process and are designed to provide specific safety analyses within each phase. Although each analysis builds on the information obtained in the SRHA, each also relies on different input documents and the required level of detail within each task grows more intense as the system approaches the final stages of development (1:22-24).

The Software Change Hazard Analysis (SCHA), Task 307, has two functions within the system safety process. First, it requires the developing contractor to analyze each and every safety-critical design change made during the development process to insure system safety is not inadvertently compromised (1:24). The second function of the SCHA begins once the system under development is delivered to the user. At this time the SCHA becomes an extension of all the 300 series tasks and provides a means in which to continue the software safety process throughout the software support phase of the system (1:24).

System Safety Program Summary. MIL-STD-882B with Notice 1 provides the current guidance for system safety requirements in weapon system acquisitions. The general system safety program requirements included in MIL-STD-882B emphasize developing a safe system. They do not differentiate between hardware and software. Specific

safety requirements are provided by the 100, 200, and 300 series tasks.

The 100 series tasks provide system safety management guidance and are the most important in establishing a system safety program. The 200 series tasks are geared toward the reduction of hardware hazards, but are often used to include software. The 300 series tasks complement the DOD software development process and represent an attempt by DOD to reduce the number of potential weapon system mishaps resulting from software errors. All of these tasks, when selectively applied and properly tailored, provide the SSM with an effective set of management tools to implement a system safety program.

Software Safety Analysis Techniques

This section examines six structured software safety analysis techniques and methodologies. Three of the six techniques, Real Time Logic, Petri Nets, and Fault Tree Analysis, are prominent in the open literature. Three additional techniques, Fault Hazard Analysis, Software Sneak Circuit Analysis, and Nuclear Safety Cross-Check Analysis were identified during a survey of aerospace industry and DOD representatives.

Many other forms of software safety analysis were identified in the literature and discussed during the survey. A list of these other techniques is included in

Appendix C. The following sections describe the six structured techniques and methodologies mentioned above.

Real Time Logic. Real Time Logic (RTL) provides the basis for a software safety analysis technique which uses a formal logic "especially amenable to reasoning about the timing behavior of systems" (23:890). This approach concentrates on the system specification rather than the final program design (23:891). The Real Time Logic analysis process begins with the system designer choosing "a model of the system in terms of events and actions" (24:142). The purpose of the "event-action model is to capture the data dependency and temporal ordering of the computational actions that must be taken in response to events in real time application[s]" (23:891). The event-action model is then "mechanically translated into Real Time Logic formulas" (24:142). An example of a Real Time Logic formula is shown in Figure 4 and interpreted as follows:

> Each occurrence of the event denoting the completion of action A happens at least c(A) time units after the corresponding occurrence of the start event, where c(A) denotes the specified execution time of action A. (23:897)

$$\forall t_1 \forall t_2 \forall i [@(\uparrow A, i) = t_1 \wedge \leq @(\downarrow A, i) = t_2] \rightarrow t_1 + c(A) \leq t_2$$

Figure 4. Real Time Logic Equation (23:897)

To complete the analysis of the system, "the RTL formulas are transformed into predictors of Presburger arithmetic with uninterrupted integer functions" (24:143). From these functions "existing [Presburger arithmetic] procedures can be used to determine if a given safety assertion is a theorem derivable from the systems specification" (23:901). If, for example, the formula in Figure 4 represented a safety assertion proven by RTL to be true, then "the system is safe with respect to the timing behavior denoted by that assertion as long as the implementation satisfies the requirements specification" (24:143). Otherwise, "the system is inherently unsafe" (6:143). Real Time Logic techniques are still being developed. While Jahanian and Mok did not include an explanation of Presburger arithmetic, they note that the mathematical process involving full implementation of Presburger arithmetic is expensive and inefficient (23:900).

Petri Nets. Petri Nets incorporate system timing requirements into the analysis of software safety (28:386). Petri nets "allow mathematical modeling of discrete-event systems in terms of conditions and events and the relationship between them" (24:143). Unique symbols are used to represent places (p), transitions (t), and inputs (i) and outputs (o) on a Petri net graph. The input functions are then studied to determine their effect on the

output of the system (28:387). An example of a Petri net graph for a railroad crossing is shown in Figure 5.

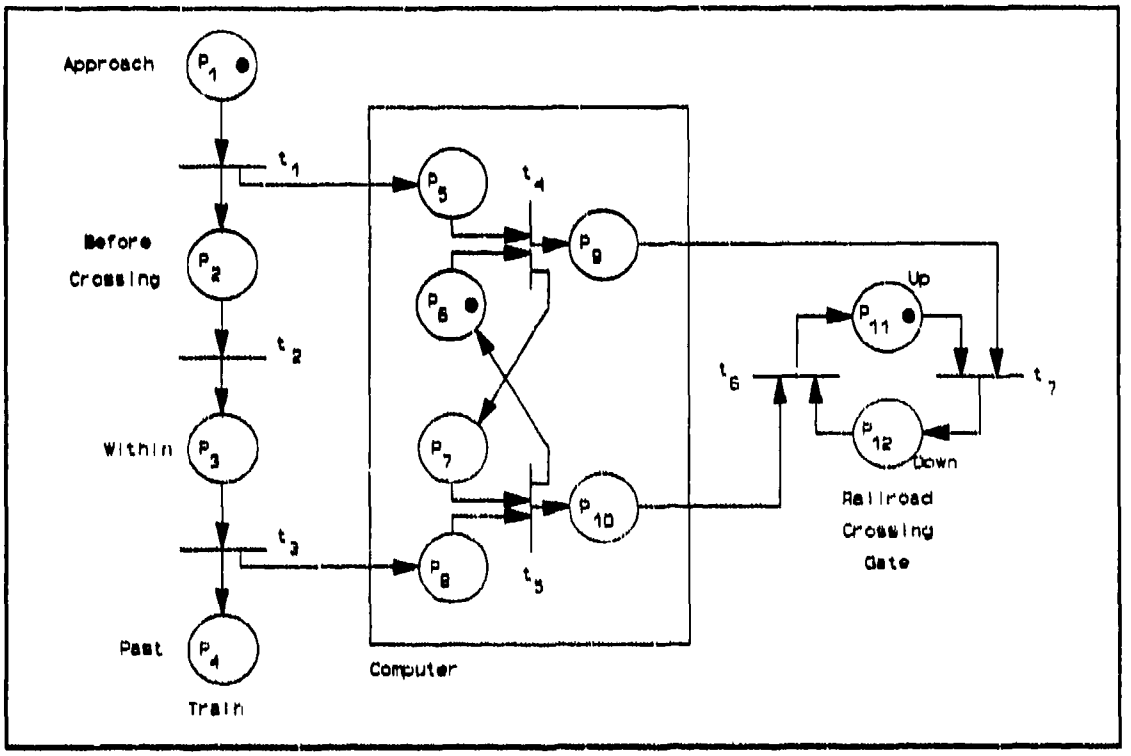

Figure 5. Petri Net Graph (28:387)

Petri net analysis is similar to Real Time Logic in that both rely on mathematical modeling of events within the system (24:143). Once the Petri net models are constructed, however, they are analyzed through simulation and mathematical techniques "to determine desirable and undesirable properties of the design" focusing on the probability of simultaneous events (24:143). The inclusion of simultaneous events is an important consideration in Petri net analysis because many safety-critical systems fail not on the basis of one failure, but usually as a result of

a series of failures associated with hardware, software, operator, or any combination of the three (33:905).

Another unique aspect of Petri nets is that potential failures can be modeled and their effects on the system studied (28:392). The modeling of potential failures can be very useful in the analysis of safety critical systems where the goal is to reduce the probability of failure by eliminating unnecessary events within the system.

Petri nets also allow the use of backward analysis procedures "to determine which failures and faults are potentially the most hazardous and therefore which parts of the system need to be augmented with fault-tolerance and fail-safe mechanisms" (24:143). Backward analysis is an important factor in the consideration of a software analysis technique because the earlier a potential failure is detected, the easier it is to modify the software requirements.

Although there are advantages associated with Petri nets as a software safety analysis tool, the development of a Petri net model for a complete system is difficult and time consuming (24:143). Computational time and computer memory requirements can easily make analysis of a complete system impractical (42:E-25). In addition, the success of the final analysis depends on the analyst's knowledge of the system and familiarity with the Petri net modeling technique (42:E-25).

Software Fault Tree Analysis. Software Fault Tree Analysis, also know as Soft Tree Analysis, is a software safety analysis technique similar to Petri nets in that it tries to "assess safety using a backward approach that starts with determining what are the unacceptable or hazardous states and then showing that these states cannot occur or that their probability of occurrence is low" (27:49). Fault Tree Analysis differs from Real Time Logic and Petri Nets in that the internal timing of the system is not considered in the analysis.

Fault trees "depict the logical interrelationships of basic events that lead to a system hazard" (4:378). Figure 6 shows an example of a complete fault tree that was used to analyze software written to control a traffic light (4:383).

The root of the fault tree in Figure 6 is the hazard; the analysis determines whether the hazard can occur due to a software failure. The lower levels of the tree are known as preconditions and are "described at the next level of the tree with either a logical AND or a logical OR relationship" (27:49). Once the fault tree has been constructed, the analysis can begin to determine whether the right combination of preconditions can be met leading to the hazard.

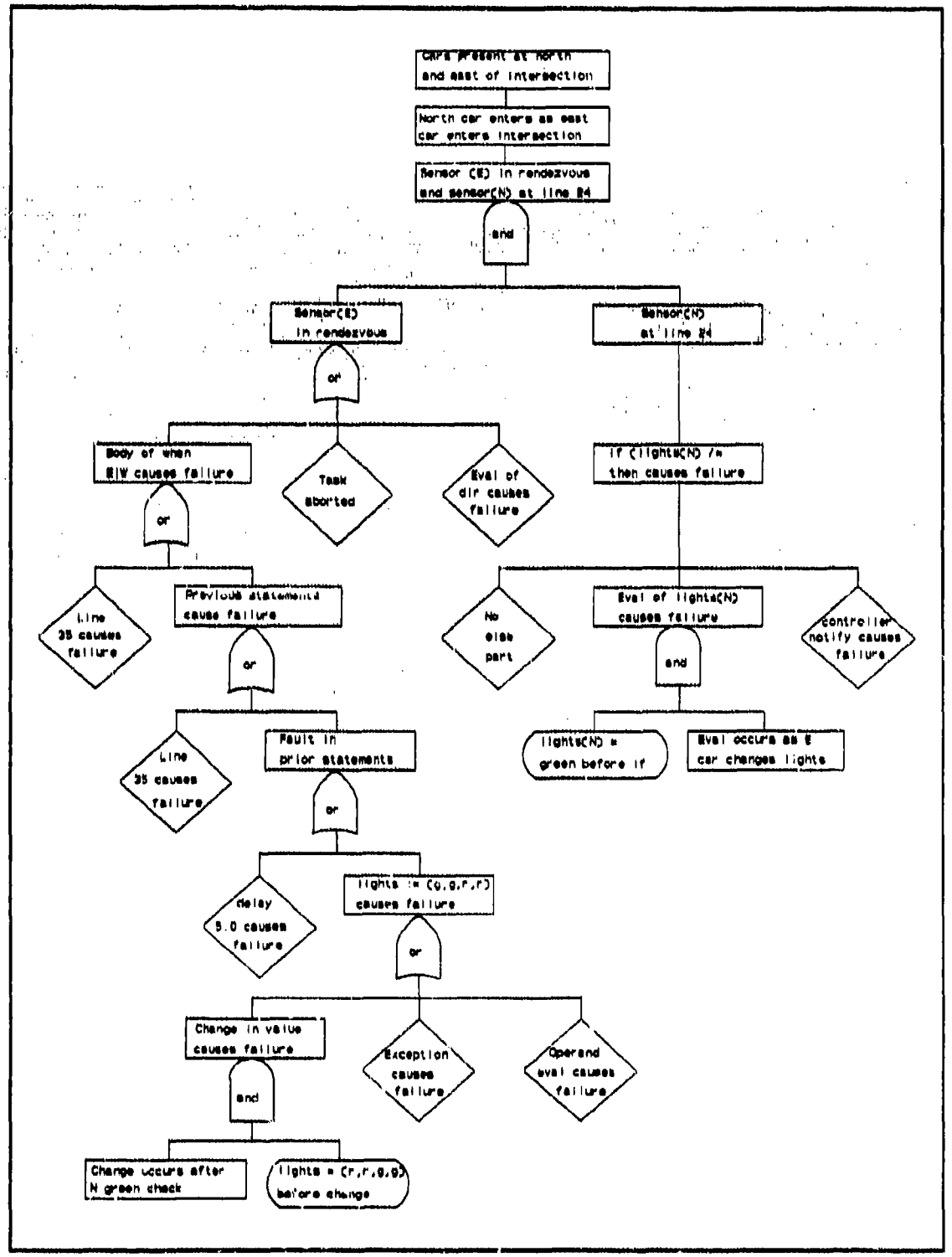

Figure 6. Software Fault Tree (4:383)

The developers of Real Time Logic have criticized fault tree analysis stating that it relies "heavily on human inspection and that fault tree analysis makes no attempt to formally capture the timing behavior of a system" (23:891). Although fault tree analysis relies heavily on human inspection, the dependence of software safety upon analysis of timing constraints has yet to be proven. Software fault tree analysis does "force the programmer or analyst to consider what the software is not supposed to do" (4:377). Although the development of fault tree analysis began nearly 30 years ago for the purpose of analyzing the safety of electromechanical systems, the technique as it applies to software is still being perfected (27:49). Software Fault Tree Analysis can be difficult to apply to a complex system and the reliability and success are dependent upon the experience of the analyst (4:386).

Software Sneak Circuit Analysis. Another software safety analysis technique adapted from hardware reliability and safety analysis is known as Software Sneak Circuit Analysis (SSCA). A report written by the Boeing Aerospace Company for Rome Air Development Center, Griffiss AFB, NY, defines Sneak Analysis as

> ... an engineering analysis tool which can be used for hardware and software systems to identify latent paths which cause concurrent unwanted functions or inhibit desired functions, assuming all component and codes are functioning properly. (3:1)

The primary objective of SSCA is to identify potential safety problems within a system's software so that they can be assessed and, if warranted, corrected before the system begins testing or is placed into operation (3:1).

The first step in the SSCA technique is to convert the program source code to a network tree representation, "such that program control is considered to flow down the page" (3:66). Difficult programs may be converted by computer, whereas simpler programs, based on the experience of the analyst, can be converted manually (3:66).

Once the topological network trees have been drawn, the next step is to identify "the basic topological patterns that appear in the trees" (3:68). There are six basic patterns shown in Figure 7. Any software program can be represented by a combination of these basic topological patterns (3:68). According to the Boeing Aerospace Report:

> The topological patterns containing branch or jump instructions have the highest incidence of problems. This includes the Return [Dome], Iteration [/Loop Circuit] and Parallel Line .... (3:68)

These patterns result in the most errors primarily because of the difficulty involved in crossing module or function interfaces within the code (3:68). Common errors are unintentional modification of memory contents, attempts to use uninitialized variables, and timing or sequencing problems (3:68).

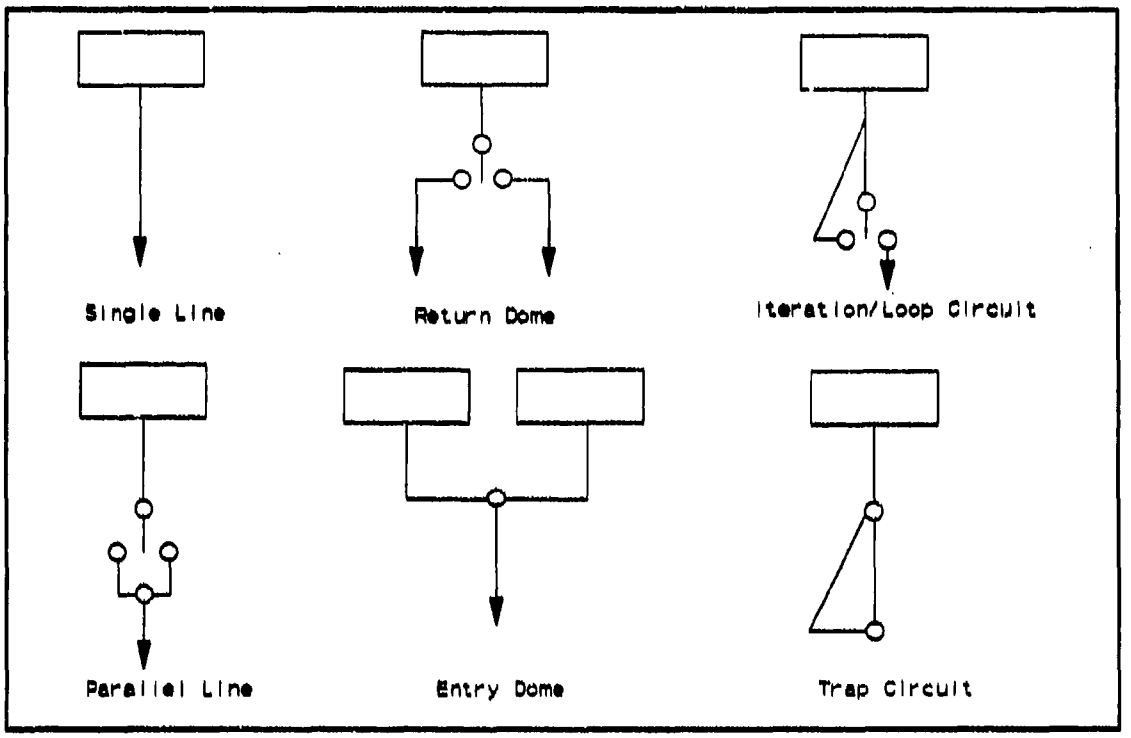

Figure 7. Software Topographs (3:68)

The next step in an SSCA is to analyze the basic structures within the topographic tree. These structures are analyzed, based on their historical characteristics, to determine whether any sneak conditions exist. Four basic types of software sneak conditions exist:

1. Sneak Output - the occurrence of an undesired output.
2. Sneak Inhibit - the undesired inhibition of an input or output.
3. Sneak Timing - the occurrence of an undesired output by virtue of its timing or mismatched input timing.
4. Sneak Message - the problem message does not adequately reflect the condition (3:68).

An identified software sneak must be verified through a complete analysis of the source code. If the software sneak is proven to be valid, "a Software Sneak Report . . . is written which includes an explanation, system level impact, and a recommendation for elimination of the sneak" (3:69).

Software Sneak Circuit Analysis is a technique that may be used "to identify system conditions that could degrade or adversely impact the mission safety or basic equipment reliability" (3:74). According to the Boeing Aerospace Company report, SSCA "should be considered for critical systems and functions where other techniques are not effective . . ." (3:74).

Although the cost of performing Sneak Analysis is high for both hardware and software, it is important to note that "regardless of the program development phase, application environment, equipment/software type, criticality ranking, or program cost, Sneak Analysis identifies a significant number of system problems" (3:141).

Fault Hazard Analysis. Fault Hazard Analysis (FHA) is an inductive analysis methodology which has its origins in hardware analysis but which can be tailored to software analysis. As such, FHA is sometimes referred to as Software Fault Hazard Analysis (SFHA) (19:21). FHA is used primarily to generate a qualitative analysis but can be extended to include some quantitative data. The FHA methodology involves the application of engineering principles and

inductive reasoning to determine "what can fail, how it can fail, how frequently it will fail, what the effects of the failure are, and how important the effects of the failure are" (10:28).

FHA is similar in form to the Failure Mode, Effects, and Criticality Analysis (FMECA) of MIL-STD-1629A. The FMECA is a reliability analysis which looks at all possible single component failures in a system. The FHA, on the other hand, must focus only on safety related failures and effects. The FMECA also has a software derivative called Software Failure Modes and Effects Analysis (SFMEA) (42:E-30). The FMECA often serves as a starting point for an FHA.

Fault Hazard Analysis appears to have its best applicability early in the design phases of a system as it requires knowledge of requirements allocation to start the effort. As the design progresses, the FHA becomes more detailed. The FHA should prepare data for use by subsequent deductive analysis techniques (11:42).

FHA is limited in that it treats software independent of hardware and can be very lengthy and tedious. The fundamental ingredient for making the FHA specifically a software safety analysis technique is its execution by a software specialist. An accurate and complete analysis requires an experienced analyst (42:E-30).

Nuclear Safety Cross-Check Analysis. A methodology developed to satisfy Air Force requirements for nuclear safety is the Nuclear Safety Cross-Check Analysis (NSCCA). NSCCA takes an adversarial approach to software analysis by employing a large number of techniques to "show, with a high degree of confidence, that the software will not contribute to a nuclear mishap" (24:145). The NSCCA process has both a technical and procedural component. The objective of the technical component is to evaluate the software by test and analysis to ensure safety requirements are met. The procedural component focuses on implementing security and control measures to "protect against sabotage, collusion, compromise, or alteration of critical software components, tools, and NSCCA results" (13:15).

The technical component of NSCCA starts with two distinct steps which comprise a criticality analysis (13:15). The first step identifies

> ... specific requirements that are the minimum positive measures necessary to demonstrate that the ... software is predictably safe according to general DOD standards for nuclear systems. (24:146)

The second step involves breaking down the software system into its lowest order functions and analyzing each function for the degree to which it controls or operates on a critical nuclear event (13:15). The degree each function influences a critical nuclear event is given a qualitative (high, medium, low) rating, and the best method to measure each function is suggested.

A program manager uses the results of the criticality analysis to develop an NSCCA program plan which establishes the required tools and facilities; required analysis; and test requirements, planning, and procedures (13:15). The independence of an NSCCA is promoted by early establishment of evaluation criteria, purpose, objectives, and expected results for specific analyses and tests (13:15).

The procedural component of NSCCA has as its objective the protection of both the software and analysis environment from any possible compromise or alteration (13:16). This objective is met by "instituting strong personnel, document, and facility security measures, coupled with stringent product control procedures" (13:16).

NSCCA has the advantage that it is usually accomplished by an agency or contractor who is independent of the software developer (13:14). In addition, it is applicable at any stage of the software development cycle (24:146). The limitation of NSCCA is that its effectiveness and expense depend entirely upon the specific analyses and test procedures selected (42:E-24).

Software Safety Analysis Technique Summary. The six software safety analysis techniques and methodologies discussed above provide a good representation of the types of software safety analyses currently available. All are considered structured in that they can be applied with some rigor to produce consistent results.

All of the techniques are expensive and time consuming when applied to complex safety-critical systems. Each relies on the analyst's capability and familiarity with the applicable analysis technique or methodology.

Although each technique can provide useful information at different phases of the development process, none can yet guarantee an absolutely safe system. Research is on-going to improve the effectiveness of each technique and to automate the safety analysis process as much as possible to eliminate analyst dependency.

Software Safety Program Summary and Conclusions

This chapter explored the relationships between the software development process, software and system safety program requirements, and software safety analysis techniques. Figure 8 shows the relationships between these three components of a software safety program. This information provides an idealized understanding of where system and software safety requirements are addressed within the software development process, what the system and software safety program requirements are, and how the software safety analysis requirements are met.

Software safety should and can be considered throughout the software development process. The software development process of DOD-STD-2167A includes numerous reviews and audits that provide ample opportunity to review software and system safety. The reviews and audits, as defined by

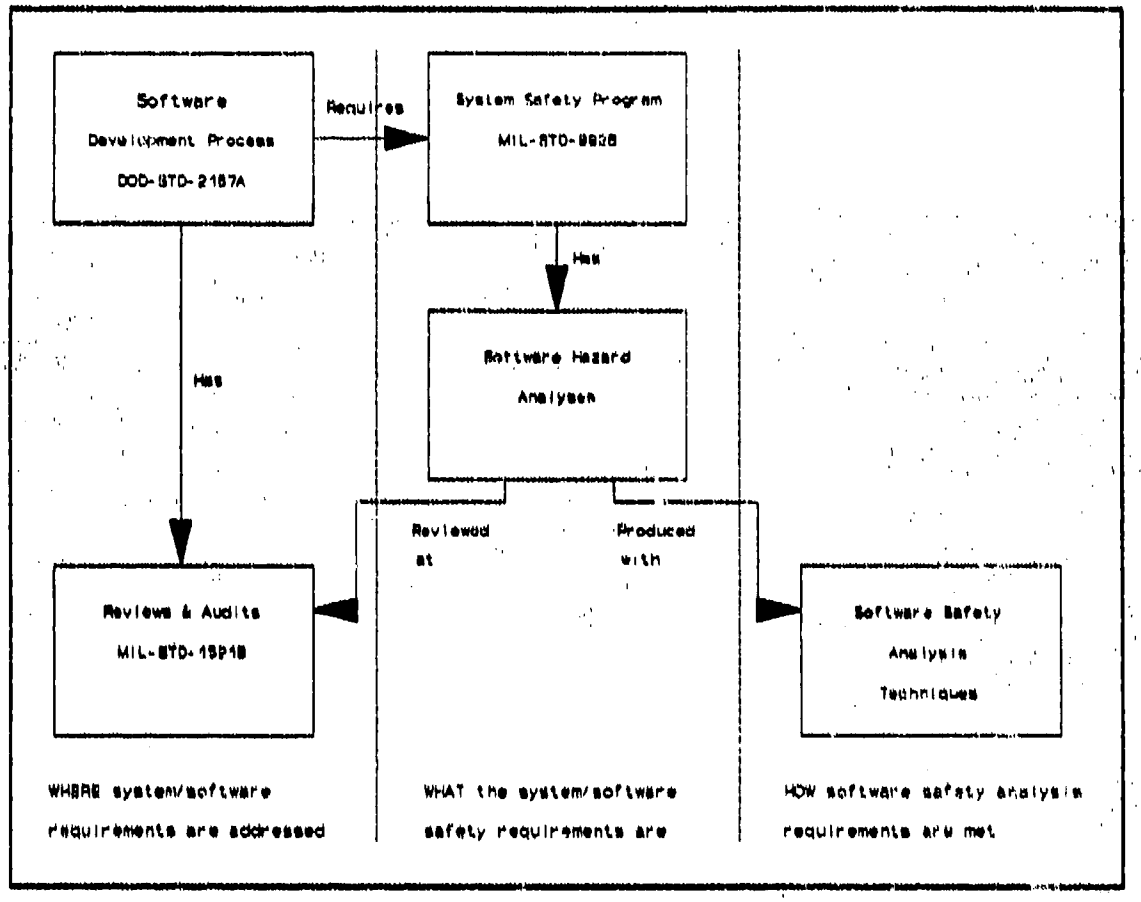

Figure 8. Software Safety Program Component Relationships

MIL-STD-1521B, each contain specific software and system safety agenda items for evaluation or review.

Specific safety program, hazard analysis, and risk assessment requirements are provided by MIL-STD-882B. The requirements tell what analyses and assessments should be performed at different phases of a development process. In addition, the requirements tell the type and reporting format of the analysis data. MIL-STD-882B does not, however, specify how the analysis data are to be obtained.

Hazard analysis and risk assessment data can be produced by structured software safety analysis techniques. There are many analysis techniques that can aid a system or software safety analysis; however, we encountered only six structured analysis techniques have been developed or adapted specifically for software safety analysis. While these structured techniques can provide the required software safety data, they are currently limited in application as well as difficult and expensive to apply.

IV. Telephone Survey

Introduction

This chapter analyzes the results of a survey of aerospace industry and DOD personnel conducted to determine what software safety analysis techniques are actually used in DOD weapon systems development and what successes or failures have been realized by their application. The four sections of this chapter discuss the characterization of the sample population, survey questions, survey responses, and conclusions.

Characterization of the Sample Population

The survey, located in Appendix D, was administered to a sample population that consisted of

- Recognized experts in software safety analysis
- System and software safety engineers and managers
- Project engineers and managers

The individuals selected for this survey represent a cross-section of professionals throughout the aerospace industry and DOD that are experienced in the system or software safety disciplines.

There were 54 individuals contacted throughout the survey process. However, only 23 were experienced in software development and/or software safety. These 23 individuals were selected to participate in this study and became the sample population.

The target organizations for this study consisted of aerospace industry and DOD representatives involved in system and software safety. Table 7 shows the break-out, by organization, of participants between the aerospace industry

Table 7. Survey Participant Organizations

	Aerospace Industry	DOD		Other	Total
		Military	Civilian		
Total Participants	8	4	6	5	23
Percent	34.8	17.4	26.1	21.7	100.0
		43.5			

and DOD. The aerospace industry participants account for 34.8 percent of the sample population and the combination of DOD military and civilian participants account for 43.5 percent of the sample population.

The "Other" classification in Table 7 consists of survey participants that did not fit into the Aerospace Industry or DOD classification. They include three DOD support contractors, one software safety consultant, and one research institute representative. As shown in Table 7, the "Other" category accounts for 21.7 percent of the sample population.

The target professions for this study consisted of system, software, and project engineers and managers. Table 8 shows the professional classification of the survey

Table 8. Survey Participant Professions

	System/Software		Project		Other	Total
	Engineer	Manager	Engineer	Manager		
Total	6	6	3	5	3	23
Percent	26.1	26.1	13.0	21.8	13.0	100.0
	52.2		34.8			

participants. The System/Software category accounts for the majority of the participants at 52.2 percent. We consider this to be an advantage because these individuals were most knowledgeable about the mechanics of applying software safety analysis techniques. The 34.8 percent in the project category provided the most information concerning the management aspects of software safety.

The 13.0 percent in the "Other" classification of Table 8 provided information concerning both the application and management aspects of software safety analysis techniques. They included one software safety consultant and two researchers.

Based on the above data we believe that each of the targeted organizations and professions is well represented in this study. Therefore, the survey results provide an accurate picture of the current state of software safety as it relates to DOD weapon system development. Additional information on each member of the sample population is included in Appendix E.

Survey Questions

The survey questions were structured as open-ended questions to gain a first-hand account of how an organization manages and conducts system and software safety programs. Typically, the survey question was just a lead-in to a more detailed discussion of software safety. Each question from the survey is presented below followed by its intended purpose.

Question 1. Do you work with the DOD software development process as defined in DOD-STD-2167A or some close derivative?

This question established survey participants' familiarity with the DOD's software development process. The question served as the first of two filters to narrow the pool of potential participants.

Question 2. Is software safety or system safety an issue that is faced in your organization?

This question established survey participants' familiarity with software and/or system safety. The question served as the second filter to narrow the pool of participants down to the sample population.

Question 3. Is software safety considered internal or external to the overall software development process?

This question assessed the relative importance of software safety as part of an organization's overall system or software development process. For this question, we

considered an internal software safety program as one that was fully integrated into the software development process and addressed safety issues throughout the process phases. An external software safety program, on the other hand, was a program not fully integrated into the software development process and only addressing safety issues at a few discrete points in the development process.

Question 4. Are there any specific software safety analysis techniques and/or management processes related to software safety employed in your organization? Explain.

This question identified the software safety techniques and methodologies used by survey participants to address software safety. This information contributed to our assessment of the degree of use and viability of structured software safety analysis techniques.

Question 5. What successes and/or failures has your organization had with software safety analysis techniques?

This question established the viability of structured software safety analysis techniques. The focus was on determining what criteria survey participants considered to be good measures of success and failure for the application of these techniques.

Question 6. In your opinion, what are the benefits or drawbacks to applying software safety analysis techniques?

This question helped identify strengths and weaknesses of the structured analysis techniques as well as management

and training implications of using the techniques. Attributes of specific techniques as well as structured analysis techniques as a group were sought.

Survey Responses

Synthesized responses to the survey questions are presented below.

Question 1. Do you work with the DOD software development process as defined in DOD-STD-2167A or some close derivative?

All survey participants working on DOD programs involving software development follow the software development process defined in DOD-STD-2167A to some degree. As expected, the development process phases for each project discussed during the survey were tailored to fit unique project situations and system requirements. Along with MIL-STD-2167A, all projects employ some form of a quality program plan as outlined in DOD-STD-2168, Defense System Software Quality Program. Some of the older contracts discussed during interviews use tailored versions of DOD-STD-2167 along with MIL-S-52779A, System Software Quality Program (the predecessor to DOD-STD-2168). Most of these older contracts, however, are close to completion or in the process of being updated to include the latest standards. The more recently awarded contracts use DOD-STD-2167A in conjunction with DOD-STD-2168.

Question 2. Is software safety or system safety an issue that is faced in your organization?

All survey participants involved in DOD weapon system development agreed that system safety is a concern that they must address in their development process. In addition, they recognized MIL-STD-882B as the principal guidance document for DOD system safety program requirements. Finally, all participants recognized that software is an important aspect of system safety and knew MIL-STD-882B contained software safety analysis tasks.

Question 3. Is software safety considered internal or external to the overall software development process?

Although all survey participants generally agreed on the importance of software safety in a system context, it was apparent that many organizations do not treat software safety as an internal part of their development process. Many participants indicated that they depended on a thorough Failure Modes and Effects Analysis and overall system safety analyses to be performed by their reliability, safety, and systems engineers. The software requirements along with relevant failure mode information are then handed to the software developers who design, code, and test the software products. If the resultant software meets design requirements both for what it is intended to do and for what it is not intended to do, then the software is considered safe. A few participants stated that if the overall system

safety is addressed throughout development, software safety could easily be verified during system testing.

Question 4. Are there any specific software safety analysis techniques and/or management processes related to software safety employed in your organization? Explain.

Most survey participants recognized Software Fault Tree Analysis and Software Petri Net Analysis as structured software safety analysis techniques. None of the participants, however, had used either of the techniques in performing software safety analyses. Many survey participants were unfamiliar with Real Time Logic and none had ever used the technique.

Survey participants identified Software Sneak Circuit Analysis, Nuclear Safety Cross-Check Analysis, and Fault Hazard Analysis as additional structured safety analysis techniques and methodologies applicable to software safety. Of these later techniques, only Fault Hazard Analysis (FHA) was widely used. In addition, many participants indicated they use various general purpose analysis and inspection tools (included in Appendix C) to aid in their analysis of software.

Two industry survey participants perform hardware based fault tree analysis as part of their overall system safety program. This analysis is intended to analyze hardware modules, but can include software by default if the software is embedded in the hardware module. Another industry

participant uses hardware based sneak circuit analysis in much the same way. Neither method, however, is specifically intended for software safety analysis.

The predominant method for performing software safety analyses appears to be Fault Hazard Analysis (FHA). FHA is used to identify potential hazards and safety-critical software in a system. Even those participants who do not have a specific software safety program also employ a form of FHA to identify software related system hazards and then provide a list of those hazards, along with a set of software requirements, to their software developers.

Survey participants using FHA were asked if there are any specific guidance and/or procedures for applying FHA methodology. Responses varied for each participant, however, a general pattern was evident. Steps common to the FHA methodology include brainstorming potential hazards, listing known hazards for similar systems, and performing probability analyses on potential hazards. No industry standard was identified for conducting FHAs, however, each participant's organization had some general guidelines. As for guidance, all participants pointed to MIL-STD-882B, System Safety Program Requirements, which contains tasks specifically called out in DOD contracts. However, survey participants pointed out that MIL-STD-882B only provides details of what specific hazard analyses should address, it

does not provide guidance as to how an analysis should be conducted.

There were no specific management processes identified to insure software safety is included in product design and development. There was however, a sense of management commitment toward the development of a safe system which includes safe software. Each project encountered has, on both the contractor and government teams, someone responsible for system safety. Contractors typically have a number of safety and system engineers involved in the system design and development while the government typically has a single System Safety Manager (SSM).

A follow-up question was asked of survey participants concerning the qualifications and training of safety personnel. Responses indicated that both industry and DOD personnel are often assigned system safety duties with minimal safety qualifications. Some of the participant's organizations have developed their own system safety training courses. These purely introductory level courses are from 1 to 2 weeks long and are aimed at project managers, SSMs, safety engineers, and system engineers. Other industry organizations require their safety personnel to obtain system safety certification through college or technical school courses. While all of these courses include some software safety discussion, none are devoted

solely to software safety or software safety analysis techniques.

A second follow-up question was asked of participants concerning their perceived need to acquire analysis capabilities with one or more of the structured software safety analysis techniques described in Chapter III. Most participants were quick to point out that they were fully capable of meeting current contractual requirements with the FHA methodology. The general consensus among contractors was why spend time and money on obtaining structured software safety analysis capability that the government has not asked for!

Question 5. What successes and/or failures has your organization had with software safety analysis techniques?

As none of the structured techniques were employed by survey respondents, there were no documented successes or failures to be discussed. Even though FHA was the predominant method used to ensure software safety, it is viewed by participants as an inductive methodology that may include a number of structured techniques. Survey participants found it impossible to quantify how successful FHA has been. It was pointed out that simply having a methodology to use is better than no analysis at all. None of the participants were able to define a measure for failed application of a technique.

Question 6. In your opinion, what are the benefits or drawbacks to applying software safety analysis techniques?

All participants agreed that the biggest benefit to be realized in the successful application of software safety analysis is a higher quality product. They also agreed that an increase in quality will lead to increased functionality, reliability, and maintainability.

There were a number of disadvantages discussed concerning the application of currently known structured software safety analysis techniques. The most common response indicated that the techniques as a group are difficult to learn and expensive to put into practice. Furthermore, the accuracy of an analysis depends almost entirely on the analyst's skill with the technique and knowledge of the system being analyzed. Finally, it was pointed out by a number of participants that these techniques need further refinement to bring them from a purely theoretical realm into a more practical application realm.

Follow-up discussions were held with a number of DOD survey participants concerning the adequacy of contractor delivered safety analyses. They indicated that there appear to be two principal problems. First, program offices typically task contractors to perform only system level safety analyses (200 series tasks in MIL-STD-882B) that treat software as a secondary issue. It was pointed out

that even though MIL-STD-882B does contain seven 300 series tasks specifically devoted to software safety analysis, program offices opt for the 200 series analyses because they are cheaper to obtain and easier to understand. Second, there appears to be a concern that even if the program offices are willing to require the 300 series analyses, there are few individuals sufficiently trained to understand or evaluate the resultant software safety analysis reports.

Survey Conclusions

There is a general awareness within the aerospace industry and DOD of the need for software safety. While survey participants understood that software plays a key role in system safety, it was disappointing to observe that many organizations do not have an integrated software safety program. There is also no standard management methodology being followed to ensure software safety is adequately incorporated into weapon system development.

The aerospace industry and DOD do not currently have any standard structured technique to perform software safety analyses. The Fault Hazard Analysis methodology is most often used, but specific steps in the methodology vary from one organization to the next. In addition, the aerospace industry typically responds to what the customer (DOD) is asking for, and in DOD weapon system development, the customer is not asking for structured software safety analysis techniques to be used. Most software safety

analyses are performed as part of a system safety analysis where software is often not adequately addressed.

The primary benefits of structured software safety analysis are increased system safety and quality. For the DOD to realize these benefits, software developers must be required to use structured software safety analysis techniques. All of the industry participants indicated they could develop the capability to apply any of the structured analysis techniques if there was a specific demand for them. The DOD, however, is not asking for the detailed software safety analyses that would require the use of structured techniques.

Even if the DOD did request the detailed software safety analyses, it is unlikely the typical DOD System Safety Manager could understand and evaluate the resultant products. The first step, therefore, to realizing the benefits of structured software safety analysis techniques is to make SSMs aware of these techniques, their benefits and limitations, and provide them with the training necessary to effectively implement and manage a software safety program.

V. Training and Education Requirements for System Safety Managers

Introduction

System Safety Managers throughout the DOD are responsible for implementing and managing system safety programs. As weapon systems become more software dependent, their corresponding system safety programs must include a larger focus on the software that will control the system. Interviews with aerospace industry safety managers substantiate the need for increased emphasis on software safety. To effectively implement and manage a system safety program, DOD System Safety Managers (SSM) must be trained in the software acquisition process and software safety procedures. Interviews with SSMs revealed that most consider themselves inadequately trained to effectively implement and manage the software portion of a system safety program.

This analysis explores SSM training and education requirements within the DOD by studying SSMs located at Aeronautical Systems Center, Wright-Patterson AFB, Ohio. Aeronautical Systems Center (ASC) SSMs are fully integrated into their system program offices and represent over 43 percent of the Air Force SSM corps. The methodology of this analysis consists of reviewing current ASC system safety training materials, system safety guidance and policy

documents, and system and subsystem hazard analysis documents. In addition, SSMs from each participating program office were interviewed for their perceptions of necessary prerequisites to becoming fully qualified SSMs.

The following ASC system program offices participated in this study:

- C-17 System Program Office
- Electronic Combat and Reconnaissance (RW) System Program Office
- F-16 System Program Office
- F-22 System Program Office

These programs are a representative sample of the types of programs SSMs may encounter. The four system program offices also account for eight SSMs, which comprise over 25 percent of the SSM force at ASC. A complete list of the documents reviewed from each system program office is included in Appendix F and a list of the corresponding SSMs interviewed is provided in Appendix E.

This analysis begins with a verification of the need for software specific training at ASC by providing a snapshot of the current hardware and software qualifications of SSMs. Next, this analysis describes the SSM's duties and responsibilities and the tools available to perform their job. Once the SSM position is described, the education and training requirements necessary to perform the job effectively are presented. Next, a list of currently available training courses that meet the training

requirements is presented. Finally, a 3-year SSM training program, encompassing both hardware and software, is recommended.

Current Status Of SSM Training

This analysis does not attempt to determine individual SSM training needs, but instead focuses on the training needs of the entire ASC SSM corps. Table 9 summarizes the current qualification and training status of SSMs at ASC.

Table 9. ASC SSM Qualification and Training Status (30)

SSMs	ASC Total	Military	Civilian	Fully Qualified		Need Training	
				HW	HW & SW	HW	SW
Number	31	22	9	12	1	19	30
Percent	100.0	71.0	29.0	38.7	3.2	61.3	96.8
HW => Hardware SW => Software							

The current ASC SSM qualification process is an ad hoc mixture of formal training courses and on-the-job training. Most of the training focuses on the hardware aspects of system safety and therefore does not produce a fully qualified SSM with software safety skills. An SSM's level of qualification is currently determined by subjective assessment and has not, historically, taken software safety into consideration.

The need for an SSM training program is evident because, as Table 9 shows, over 61 percent of ASC SSMs are not fully qualified in hardware oriented system safety, and over 96 percent are not fully qualified in software system safety. Although the hardware oriented training need is large, the need for software safety training is overwhelming. Interviews with SSMs indicate the training shortfall is almost entirely in the areas of the software development process and software safety assurance methods. This shortfall represents a significant near term training burden.

Current ASC estimates indicate it takes approximately 3 to 5 years to fully qualify a new SSM assuming no prior system safety experience (30). Table 9 shows that military SSMs make up 71 percent of the SSM force at ASC while civilian SSMs comprise only 29 percent of ASC's SSM corps. Civilian SSMs, however, remain in their jobs an average of three to four times longer than their military counterparts and as a result, represent a much smaller training burden (30). Military SSMs, on the other hand, are assigned for only 3 to 5 years and will most likely not work as SSMs in their subsequent assignments. The turnover rate for military SSMs at ASC is approximately five per year (30); these turnovers represent the bulk of the long term training load and support the need for an on-going, consolidated

training program as the primary means of maintaining fully qualified SSMs.

The SSM's Job

Before a training and education recommendation can be made, the requirements of the SSM's job need to be understood. The following SSM job description is based on our evaluation of ASC local training materials, policy and guidance documents, and personal interviews with SSMs.

According to ASC System Safety training materials, system safety management is defined as the process that

> ... provides the optimum degree of safety within the constraints of operational effectiveness, schedule, and cost through timely application of system safety management and engineering principles whereby hazards are identified and corrective actions applied to minimize [safety] risk .... (29)

The Program Manager's System Safety Guide indicates that the role of the SSM is to provide a competent and responsible "safety overview of the technical and management aspects of the entire program" (10:33). Therefore, the SSM is the primary point of contact for all system safety matters, management and technical, within a system program office.

The SSM's day-to-day job consists of many specific duties which are listed in references (10:7), (11:25-27), and (29). The duties involve both management and technical tasks and require the SSM to maintain a continuous awareness of system safety program requirements, design and development issues, and contractual issues affecting the

safety of the system. Management duties include

- Remaining abreast of system safety policies and regulatory requirements.
- Reviewing program management documents for inclusion of relevant system safety requirements.
- Ensuring appropriate system safety requirements and tasks are included in procurement documents.
- Determining what safety critical issues will be addressed at program reviews and what contractual changes may be required to address emerging safety-critical items.

Technical duties include

- Reviewing and evaluating the contractor's System Safety Management Plan for adequacy of system safety analysis, design, and test procedures.
- Evaluating and monitoring contractor performance in system safety analysis, design, and test.
- Initiating system safety programs to ensure safety hazards are eliminated or sufficiently controlled.

While it is the system program manager's ultimate responsibility to ensure that the safest possible system is developed, the program manager provides the SSM with the necessary authority to perform the system safety management functions (10:33). The SSM, in turn, performs the day-to-day system safety management functions and keeps the program manager informed of relevant system safety issues.

The SSM's Tools

There are many tools available to the SSM to assist in performing system safety related responsibilities. These

tools include

- The SSM's training and education.
- System safety guidance and policy documents.
- System safety analysis and risk assessment reports.
- Program reviews and system safety working groups.

The most important tool that an SSM possesses is a good education and training background. The SSM works with many people having both technical and management interests and must project a self-confident and credible image. While many personal qualities can contribute to credibility, SSMs must possess a thorough knowledge of their job that comes only from having the proper education and training.

System safety guidance and policy documents tell the SSM what must be done to insure development of a safe weapon system. These documents form the basis for system safety requirements which must be levied on the developing contractor. Contractor compliance with safety requirements and the elimination or control of safety hazards can be verified by the SSM through evaluation of system safety analysis and risk assessment reports. System safety managers coordinate with system and specialty engineers within the program office to verify the technical accuracy of the analyses and reports. Program reviews and working groups provide the SSM with feedback to gauge the status of system safety efforts as well as offering a forum for face-to-face problem identification and resolution.

The SSM's Training and Education

The training and education requirements for SSMs were derived by reviewing safety hazard analysis and risk assessment reports, interviewing SSMs, and considering the SSM job description from the previous section. The training requirements for an SSM fall into two categories. First, there are requirements that involve general system and safety related knowledge. They represent skills that apply to the system safety management of any system regardless of its mix of hardware and software. Second, there are training requirements directly related to software development and its impact on system safety. SSM training requirements are shown in Table 10 as an enumerated list to facilitate cross-referencing to detailed requirements discussion and available training.

Review of hazard documents revealed technical skills that an SSM should have knowledge of in order to understand and evaluate hazard analyses and risk assessments. As a whole, the document review was extremely helpful in identifying methodologies, skills and analysis techniques relating to system safety. The treatment of software safety, however, in the documents reviewed was disappointing. None of the documents included a detailed software safety analysis. One analysis virtually dismissed software as a system safety issue while another merely included a few generic software design issues taken from a

Table 10. SSM Training Requirements

Training Requirements
1. System Acquisition Process
2. System Development Process
3. System Safety
a) Guidance and Policy
b) Management
c) Analysis
d) Design
e) Testing
4. Risk Assessment
5. System Understanding
6. Software Acquisition Life-Cycle
7. Software Development Process
8. Software Test and Evaluation
9. Software Safety Analysis Techniques

software development handbook. However, the Software Requirements Hazard Analysis obtained from the Electronic Combat and Reconnaissance System Program Office employed the FHA methodology to provide an initial look at the safety hazards associated with software requirements.

Interviews with SSMs corroborated the technical training requirements and provided the primary means for determining management skills an SSM should have in order to effectively manage a system safety program. The degree of understanding required for any specific training area was

determined by comparing specific skills and principles to the SSM job description. Finally, the educational background of an SSM was determined by assessing the specific skills and principles encountered in the training requirements.

Training Requirements. An SSM should have training in each of the following general areas:

1. System acquisition process. An SSM should understand the basic mechanics of system acquisition at both the DOD and program office levels. At the DOD level, an SSM should understand the major milestones in the acquisition process and know how a program evolves from one milestone to the next. At the program office level, an SSM must understand day-to-day acquisition functions such as development of requirements and specifications; evaluation of contractor proposals and plans; and reviewing and approving contractor submitted data. Interviews with SSMs indicate that knowing where a system is in its particular life-cycle and knowing what details that stage of the life-cycle entails allow the SSM to address system safety at an appropriate level. Additionally, the SSM who understands program office acquisition functions can become a competent and trusted member of the program office team and impact system development with appropriate system safety requirements.

2. System development process. An SSM should know how a system is developed and understand how a system developer goes through the process of requirements analysis, making engineering trade-offs, designing, building, and testing. Additionally, the SSM should understand how hardware and software development efforts differ and how their eventual integration into a system affects the safety of the overall system. SSM interviews indicated that understanding the development process is vital to an SSM's ability to evaluate and direct the contractor.

3. System safety. The SSM should thoroughly understand how to implement and manage a system safety program. An SSM needs training in the areas of safety guidance and policy, management, analysis, design, and testing.

a. Guidance and Policy. The SSM must know which directives, policies, regulations, and handbooks provide guidance and requirements on system safety issues. An SSM must be able to evaluate the precedence, applicability, and system impacts of the many system safety guidance and policy sources. Interviews with SSMs indicate that knowing system safety guidance and policy allows the SSM to place the proper requirements on contract.

b. Management. The SSM must be able to manage informally as well as apply formal system safety management tools. The SSM needs to understand MIL-STD-882B

fully and be able to select appropriate tasks to plan, conduct, and evaluate a system safety program. Interviews with SSMs indicate that an SSM must be able to review and critique the System Safety Program Plan (SSPP) as well as any of the hazard analyses and assessments employed on a particular development effort.

c. Analysis. An SSM should be familiar with system safety analysis techniques used by industry. A rudimentary understanding of safety analysis techniques is required to evaluate and critique contractors' safety analyses. This requirement is supported by the fact that all of the hazard analysis reports reviewed employed some form of Fault Hazard Analysis (FHA) to address software safety. In addition, the F-16 System Safety Hazard Analysis employed a detailed hardware based Fault Tree Analysis (FTA). The F-22 analyses selectively employed FTA in determining event probabilities. Finally, the C-17 analyses employed FTA and Sneak Circuit Analysis on selected safety critical subsystems.

d. Design. The SSM should be familiar with safety design techniques and methodologies. SSMs should be aware of how safety risks posed by a particular weapon system can be avoided or controlled to an acceptable level.

e. Testing. The SSM should understand both the development and operational test and evaluation

processes for weapon systems. An SSM needs to understand

- the limits of testing;
- formulation of testing requirements;
- and principles of validation and verification.

System safety guidance documents indicate that ". . . safety requirements need to be verified by analysis, inspection, demonstration, or test" (8:A-15).

4. Risk assessment. The SSM should understand how system safety risk assessments are performed as well as have a basic understanding of the quantitative analysis of event probabilities. In addition, the SSM needs to understand how system reliability is determined and how reliability information is integrated into the system safety risk assessment.

5. System understanding. An SSM must be familiar with the system under development. Every system safety analysis report reviewed starts with a system description which serves as the basis for the analysis. Interviews with SSMs indicate a working knowledge of the individual subsystems that comprise a weapon system as well as the integrated whole weapon system is needed.

6. The software acquisition life-cycle. The SSM should understand the characteristics of the software life-cycle in terms of software development and maintenance. Understanding that software does not break or fail in the traditional hardware sense is important in determining the

reliability and performing failure mode analysis of a software intensive system.

7. The software development process. An SSM must understand how software is designed, built, and tested. Specific knowledge of the design, code, testing, and redesign cycle for a Computer Software Unit, Computer Software Component, and Computer Software Configuration Item development will aid the SSM in determining how safety impacts the software development process.

8. Software test and evaluation. The SSM must be familiar with basic approaches to software testing and evaluation. The SSM must understand that exhaustive testing of software is virtually impossible, and therefore, the need for careful design and analysis of software testing is critical to ensuring software system safety.

9. Software safety analysis techniques. The SSM must stay up to date on current software safety analysis techniques as well as techniques that are under development. The SSM must be in a position to lead the contractor and suggest possible software safety analysis techniques as there is currently no standard method to perform software system safety analysis and assessment within the DOD and industry.

Education Requirements. Based on the levels and types of training identified above, the minimum education requirement for an SSM should be a bachelor of science

degree in engineering or some related applied science. To evaluate and direct the work of both system and safety engineers, the SSM should be conversant in both the language and techniques of the scientific community.

Based on the above training and education requirements we believe the ideal SSM should be a technically oriented person with at least a basic understanding of the DOD acquisition development process (hardware and software); system and software safety guidance and policy documents; and, as governed by their program, the applicable hardware and software safety analysis techniques in use by their contractor.

Available Training

The best way for an SSM to become fully qualified is through the proper combination of formal training courses and on-the-job training. The following list presents the available training courses that can help meet the formal training needs of an SSM.

ASC Training Course:

1. ASC System Safety Training Course - conducted locally by the ASC System Safety Office

Air Force Sponsored Courses:

2. WSYS 100 - Introduction to Acquisition Management (or its equivalent SAS 001)

3. WSYS 200 - Acquisition Planning and Analysis (or its equivalent SAS 006)

4. SAS 002 - Computer Resources Acquisition Course

5. WSYS 212 - Mission Critical Computer Software Project Management

6. WSYS 229 - Test and Evaluation Management

7. WCIP 057 - Systems Safety Management Course

8. WCIP 060 - System Safety Analysis Course

9. WCSE 471 - Software Engineering Concepts

10. WCSE 472 - Specification of Software Systems

11. WCSE 473 - Principles and Applications of Software Design

12. WCSE 474 - Software Generation and Maintenance

13. WCSE 475 - Software Verification and Validation

14. WQMT 020 - Reliability and Maintainability Overview

15. WQMT 335 - Reliability and Maintainability Design in Systems Acquisition

Navy Sponsored Courses:

16. SS-320 - Systems Safety Concepts

17. SS-510 - Systems Safety and Human Factors

18. SS-520 - Advanced Systems Safety Techniques

Commercially Available Courses:

19. System Safety Course - available through the University of Southern California

20. Software Safety Course - available through the University of Southern California

21. Software Safety Workshop - available through the University of California-Los Angeles

Evaluating the above courses helped to determine which training requirements are met by each course. The

evaluations were based on course descriptions contained in the 1 September 1991 version of AFR 50-5, Training: USAF Formal Schools, current course syllabi, and interviews with SSMs. For more information on the courses listed above refer to Appendix H.

The results of the course to requirements evaluations are compiled in Table 11 and Table 12. An 'X' in the table indicates that the minimum training requirement is met. Although several of the requirements are met by numerous courses, it is the opinion of the researchers that each subsequent course (within a sequence) builds on concepts learned in the previous course and further enhances subject understanding.

All of the training requirements listed in Table 11 and Table 12, except system understanding, can be met by several course combinations. System understanding, however, is a requirement that is conceptually different for each SSM. It relates to the particular weapon system or subsystem under development and is one of the most important aspects of an SSM's job. SSMs must assume the responsibility to become familiar with their systems. Formal training cannot compensate for a lack of system understanding.

Table 11. Training Requirements Matrix

Training Requirements	Air Force Sponsored Courses														
	ASC LCL	WSYS				SAS 002	WCIP		WCSE					WQMT	
		100	200	212	229		057	060	471	472	473	474	475	020	335
1. System Acq Process	X	X	X	X	X	X	X		X			X		X	
2. System Dev Process	X	X	X	X	X	X	X		X	X	X	X	X		
3. System Safety	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
a) Guidance & Policy	X														
b) Management	X						X	X						X	
c) Analysis	X						X	X						X	X
d) Design							X	X						X	X
e) Testing							X	X						X	X
4. Risk Assessment	X			X	X		X	X	X		X	X	X	X	X
5. System Understanding															
6. Software Acq Life-Cycle	X	X	X	X	X	X	X		X	X	X	X	X		
7. Software Dev Process	X	X	X	X	X	X	X		X	X	X	X	X		
8. Software T & E		X	X	X	X	X	X	X	X	X	X	X	X	X	X
9. SSA Techniques							X	X							

Table 12. Training Requirements Matrix (Continued)

	Navy Sponsored Courses			Commercially Available Courses		
				USC		UCLA
Training Requirements	SS-320	SS-510	SS-520	System Safety	Software Safety	Software Safety Workshop
1. System Acquisition Process						X
2. System Development Process				X		X
3. System Safety	-	-	-	-	-	-
a) Guidance & Policy				X		X
b) Management				X	X	X
c) Analysis	X	X	X	X	X	X
d) Design		X	X	X	X	X
e) Test			X			X
4. Risk Assessment	X	X	X	X	X	X
5. System Understanding						
6. Software Acquisition Life-Cycle				X		X
7. Software Development Process				X		X
8. Software T & E						X
9. SSA Techniques	X	X	X	X	X	X

Recommended Training Program

SSMs are responsible for making safety-critical decisions relating to system and software safety that may prevent loss of life or damage to equipment. Many of these decisions are based on the review of hazard analyses and risk assessments submitted by the developing contractor. To effectively evaluate the analysis reports an SSM must be adequately trained.

As discussed in the previous section, both short and long term SSM training needs must be met. Many of the SSMs interviewed think they would benefit from additional training, specifically in the area of software development and safety analysis. In addition, there is a consensus among SSMs on the need for an introductory level consolidated training program for newly assigned SSMs.

To address the need for a consolidated training program that provides an SSM with a complete system (including both hardware and software) safety management capability, a training program including the curriculum listed in Table 13 is recommended. The program outlined represents a minimum set of existing courses and involves a number of assumptions concerning the trainee, course content, and the training environment.

The best way to outline a comprehensive training program is to assume the trainee is a newly hired individual with only an undergraduate degree in engineering or applied

Table 13. Recommended Training Curriculum

Course Number	Title
ASC Local Course	Introduction to System Safety and Acquisition Management
WSYS 100	Introduction to Acquisition Management
WSYS 200	Acquisition Planning and Analysis
SAS 002	Computer Resources Acquisition Course
WCIP 057	Systems Safety Management Course
WCIP 060	System Safety Analysis Course
WCSE 471	Software Engineering Concepts
WQMT 020	Reliability and Maintainability Overview
SS 510	Systems Safety and Human Factors
SS 520	Advanced Systems Safety Techniques
SFT YY-#	Software Safety Course @ USC
ENG 819.230	Software Safety Workshop @ UCLA

science. Since not all SSMs come to the job void of safety related experience, the recommended program can be easily modified to accommodate individual training needs.

Based on the course to requirements evaluations conducted previously, the training courses included in this program meet or exceed the minimum training requirements. Other courses listed in Table 11 and Table 12 may be added to meet a specific training need or to enhance an SSM's general knowledge.

The training environment is assumed to be ideal in that required courses, training time, and training funds are

available when needed. These basic assumptions allow us to present the following training program.

Figure 9 shows a 3-year training program which includes all courses listed in Table 13. The courses are scheduled to insure that prerequisites will be met. The schedule also shows liberal amounts of slack time between courses to

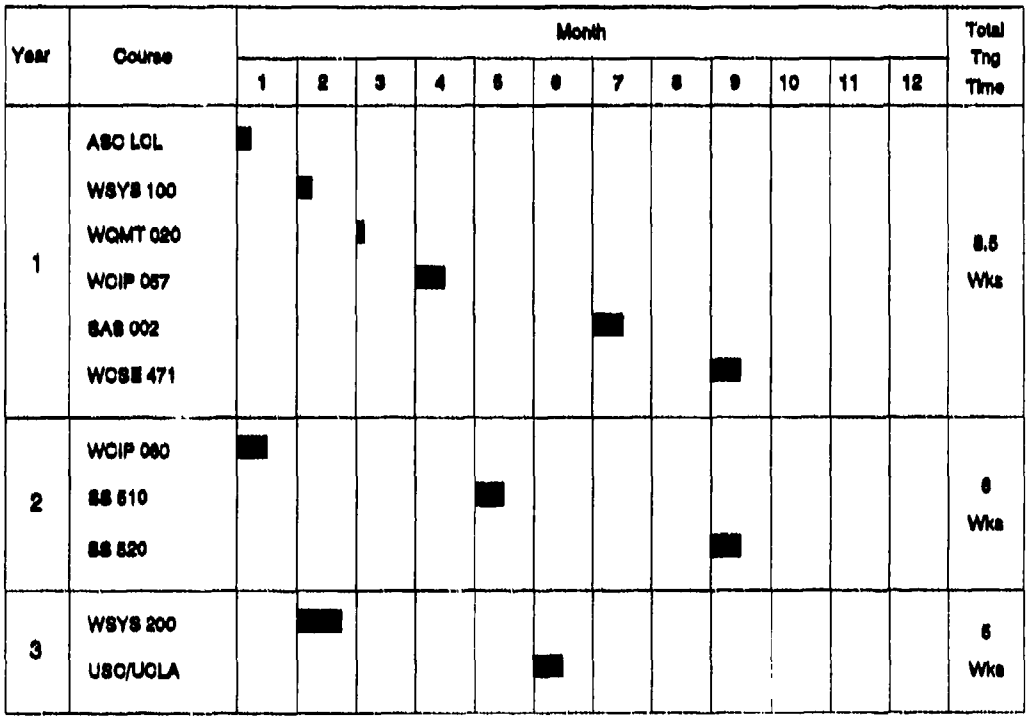

Figure 9. Recommended SSM Training Program

conduct on-the-job training or assimilate course material. The allocation of slack time in Figure 9 is purely discretionary. The 3-year schedule allows for a mix of formal and on-the-job training. As the SSM gains

operational experience, advanced software safety management skills are presented through formal training courses.

The first year of training is designed to produce a functionally qualified SSM who can perform with minimum supervision. The ASC local system safety course provides an introduction to system acquisition, system development, and system safety. WSYS 100 provides a more detailed look at system acquisition and development. WQMT 020 provides an overview of system reliability and maintenance. WCIP 057 addresses system engineering and management and provides a more detailed look at system safety management methods. SAS 002 addresses software acquisition, the software development process, and an introduction to the Ada programming language. Finally, WCSE 471 provides a look at software engineering and development from requirements analysis to support and maintenance.

The second year emphasizes system safety analysis methods. WCIP 060 provides an introduction to software and hardware safety analysis techniques, probability theory, and risk assessment. SS-510 introduces system safety and human factors engineering techniques. SS-520 provides a detailed look at system safety analytical techniques including software safety analysis techniques.

Finally, the third year of training completes the course work and on-the-job training necessary to produce a fully qualified SSM. WSYS 200 provides detailed training on

program planning, executing, and controlling. As a final course, either the USC course or the UCLA workshop should be taken to provide an understanding of current software safety analysis trends in industry and academia.

We believe the above training program is realistic and achievable because it is based on existing and available training courses. The training program is designed to produce a functionally qualified SSM in one year and a fully qualified SSM by the end of the third year. Training time and qualification ratings are based on the assumption that the SSM has no prior acquisition or safety experience.

Conclusion

This analysis examined the education and training requirements necessary for SSMs to perform their job effectively. By examining the functions of the SSM position it is evident that the SSM is an integral part of the system development team. However, the education and training required to qualify an SSM, to date, have been predominately hardware oriented. SSMs must have system and safety knowledge that includes both hardware and software if they are going to accurately and effectively manage a system safety program. As software comes to dominate the acquisition of high technology weapon systems, training that will allow SSMs to effectively manage hardware and software safety programs must be provided.

VI. Conclusions and Recommendations

Introduction

The preceding examination of software safety in DOD weapon system development established the relationships between the DOD software development process, DOD system safety program requirements and software safety analysis techniques. The combined results from the literature review, telephone survey, and personal interviews then pointed out how these three components combine to provide some level of assurance that the system under development will operate at an acceptable level of risk.

Given the complexities of the interrelationships among the components of a software safety program, we analyzed the training and education requirements necessary to prepare DOD System Safety Managers (SSM) to implement and manage a system safety program effectively. A consolidated training program was developed and presented in Chapter V.

The next section of this chapter discusses conclusions reached following the accomplishment of the stated research objectives. Recommendations that should be implemented to enhance the DOD's position in the field of system and software safety follow. Finally, recommendations for further study are presented.

Conclusions

The following conclusions were reached as a result of accomplishing the research objectives for this study.

1. The framework for implementing and managing an effective DOD weapon system software safety program exists. The software development process defined in DOD-STD-2167A is detailed and highly structured. MIL-STD-1251B defines formal reviews and audits that provide ample opportunity to discuss specific system and software safety issues. In addition, MIL-STD-882B defines system and software safety analysis requirements. Finally, structured analysis techniques have been developed or adapted specifically for software safety analysis and can provide important software hazard information when applied throughout the software development process.

2. In practice, software safety programs are not implemented and managed effectively. The principal limitation of the software development process, from a safety standpoint, is that potentially all of the safety review can be tailored out of the process. System safety program requirements are also allowed to be tailored by DOD program managers. Often, the system-level safety analysis (200 Series) tasks of MIL-STD-882B are chosen over the more specific software safety analysis (300 Series) tasks. In response, developers use hardware oriented safety analyses that inadequately address software safety. In addition,

many development organizations have not integrated software safety into their overall software development process. Finally, there is no known standard management methodology for software safety in industry or the DOD.

3. Structured software safety analysis techniques are not widely used throughout the aerospace industry and DOD. The most common analysis approach is the Fault Hazard Analysis methodology. This is an inductive analysis method that results in a qualitative safety analysis. Most other techniques are deductive in nature and share the common characteristics of being difficult and expensive to apply. In addition, the application of these techniques has been limited to small software systems. Finally, DOD has not placed a demand on industry to apply any of these techniques and the general consensus among developers was why spend time and money on obtaining structured software safety analysis capability that the government has not asked for!

4. The Air Force needs a consolidated training program for System Safety Managers that covers both hardware and software system safety. Over 96 percent of Aeronautical System Center's System Safety Managers are not fully qualified in software system safety. SSMs are responsible for implementing and managing system safety programs that address both hardware and software safety. Currently, few SSMs fully understand software development and software

safety analysis and their training has been both ad hoc and hardware oriented.

Recommendations For Implementation

Based on the conclusions of this study, the following recommendations will increase the DOD's effectiveness in implementing and managing system and software safety programs.

1. To provide Air Force SSMs with the knowledge necessary to effectively implement and manage a comprehensive system safety program, the training program presented in Chapter V should be implemented.

2. Many aerospace contractors conduct in-house system and software safety training programs for their engineers and managers. DOD system safety engineers and managers should attend these training programs to increase their familiarity with the safety techniques and methodologies being used to develop their systems.

3. To provide Air Force engineers and managers with system and software safety familiarity, the Air Force Institute of Technology should include in its advanced degree and professional continuing education curriculums a block of instruction covering system and software safety. Target curriculums should include computer engineering, software engineering, systems engineering, systems management, software systems management, and information resource management. System and software safety instruction

should provide familiarity with current DOD system safety policy and guidance and the relationships between the development process, system safety program requirements, and safety analysis techniques.

4. To help establish a standard management methodology for software safety programs, the Air Force should implement a command-level System and Software Safety Working Group. This group should meet biannually to discuss system and software safety implementation and management issues in regards to weapon system development. This group can also provide a forum for MIL-STD-882 evaluation and establish a system and software safety lessons learned database.

5. DOD system safety engineers and managers should actively participate in industry forums for system and software safety. The Institute of Electrical and Electronics Engineers (IEEE) Computer Assurance (COMPASS) Conference and the Electronics Industry Association (EIA) are two prominent forums that focus on software safety. Presenting papers and participation in panel discussions at these forums provides an excellent opportunity for DOD representatives to influence industry's refinement of old, or development of new, software safety analysis techniques and methodologies.

Recommendations For Further Study

Based on the conclusions reached in this study, the following recommendations for further study will increase

the DOD's knowledge in implementing and managing system and software safety programs.

1. Study System Safety Manager jobs in the Army, Navy, and other government agencies. The objective would be to learn what qualification and training requirements, training programs, and training courses are used by these agencies and how the Air Force and DOD could benefit from them.

2. Develop a prototype database of software system safety hazards and risks that can be used by system and safety engineers to start a software safety analysis effort. There are numerous lessons learned databases and design handbooks throughout the DOD that address safety items. They are, however, not integrated. This task would glean software safety hazards, risks, and guidelines from various sources to be cataloged in a database.

3. Select a structured software safety analysis technique and apply it to a software module of reasonable size. Items to be studied might include what safety analysis data must be known to start the structured analysis, how hard or easy is the technique to learn and apply, and how large can the software module be before the analysis becomes intractable.

4. Track and evaluate the hazard analyses and risk assessments resulting from the 300 series tasks of MIL-STD-882B currently on contract. This study should focus on the specific techniques and methodologies used to conduct

the tasks and measure the effectiveness of the analysis obtained.

In addition to the above recommendations for further study, the following topics for study, although not directly supported by this research, can expand the DOD's knowledge of system and software safety management.

5. Track the application and resulting software safety analyses of MIL-STD-882C. MIL-STD-882C is due for distribution in December 1992 and will combine the 200 and 300 series tasks of MIL-STD-882B into a single set of system safety tasks.

6. Broaden the focus of this study by investigating more diverse applications of software system safety techniques in such fields as medical equipment development, nuclear weapons development, nuclear power systems, mass transit systems, etc.

7. Expand the understanding and practice of software safety analysis in the United States by studying how other countries like Great Britain, Australia, and Japan ensure the safety of their software systems.

Appendix A: Glossary of Terms

Acronyms

AFMC - Air Force Materiel Command

ASC - Aeronautical Systems Center

CCB - Change Control Board

CDR - Critical Design Review

CDRL - Contractor Data Requirements List

CSC - Computer Software Component

CSCI - Computer Software Configuration Item

CSU - Computer Software Unit

DOD - Department of Defense

FCA - Functional Configuration Audit

FHA - Fault Hazard Analysis

FMECA - Failure Mode, Effects, and Criticality Analysis

FQR - Formal Qualification Review

FQT - Formal Qualification Testing

FSED - Full Scale Engineering Development

GFE - Government Furnished Equipment

GFP - Government Furnished Property

LANTIRN - Low Altitude Navigation and Targeting Infrared for Night

MOA - Memorandum of Agreement

NSCCA - Nuclear Safety Cross-Check Analysis

O&SHA - Operating & Support Hazard Analysis

PCA - Physical Configuration Audit

PDR - Preliminary Design Review

PHA - Preliminary Hazard Analysis

PHL - Preliminary Hazard List

PMD - Program Management Directive

PMP - Program Management Plan

RTL - Real Time Logic

RW - Electronic Combat and Reconnaissance

SCHA - Software Change Hazard Analysis

SDF - Software Development Folder

SDR - System Design Review

SFHA - Software Fault Hazard Analysis

SFMEA - Software Failure Modes and Effects Analysis

SHA - System Hazard Analysis

SIP - System Improvement Program

SOW - Statement of Work

SPO - System Program Office

SPS - Software Product Specification

SRA - Software Requirements Analysis

SRHA - Software Requirements Hazard Analysis

SRR - System Requirements Review

SSCA - Software Sneak Circuit Analysis

SSG - System Safety Group

SSHA - Subsystem Hazard Analysis

SSM - System Safety Manager

SSPP - System Safety Program Plan

SSR - Software Specification Review

STD - Software Test Description

SWIM - System Wide Integrity Management

TRR - Test Readiness Review

UCLA - University of California - Los Angeles

USC - University of Southern California

Definitions

Base line. A configuration identification document or a set of such documents formally designated and fixed at a specific time during a CI's life cycle. Base lines, plus approved changes from those base lines, constitute the current configuration identification. For configuration management there are three base lines, as follows:

b. Allocated base line. The initial approved allocated configuration identification.

a. Functional base line. The initial approved functional configuration identification.

c. Product base line. The initial approved or conditionally approved product configuration identification. (6:61)

Hazard. A set of conditions (i.e., a state) that can lead to an accident given certain environmental [operating] conditions. (25:223)

Hazard Probability. The aggregate probability of occurrence of the individual hazardous events that create a specific hazard. (8:2)

Hazard Severity. An assessment of the worst credible mishap that could be caused by a specific hazard. (8:2)

Mishap. An unplanned event or series of events that leads to an unacceptable loss such as death, injury, illness, damage to or loss of equipment or property. (8:2)

Safety-Critical. Those software operations that, if not performed, performed out-of-sequence, or performed incorrectly could result in improper control functions (or lack of control functions required for proper system operation) which could directly or indirectly cause or allow a hazardous condition to exist. (13:3)

Software Reliability. The probability that the software will execute for a particular period of time without a failure, weighted by the cost to the user of each failure encountered. (13:3)

Software Safety. The application of system safety engineering techniques to software in order to insure and verify that the software design takes positive measures to enhance system safety and that errors which could reduce system safety have been eliminated or controlled to an acceptable level of risk. (13:3)

Software Safety Analysis. A hazard evaluation technique which identifies software commands (with and without errors) either singularly or in combination with hardware responses that could result in safety critical hazards. (13:3)

Software Security. The degree to which a piece of software protects itself, or is protected, from unauthorized, and sometimes malicious, actions. (32:Appendix 1)

System Safety. The application of engineering and management principles, criteria, and techniques to optimize safety within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle. (8:3)

Validation. The process of confirming that the software (i.e, documentation and computer program) satisfies all user requirements when operating in the user environment. (12:C-10)

Verification. The process of confirming that the products of each software development phase (e.g., requirements definition, design, and coding) are complete, correct, and consistent with reference to the products of the preceding phase. (12:C-10)

Appendix B: Guidance and Policy Documents

This appendix lists guidance and policy documents applicable to the DOD software development process and system safety program. This list is not exhaustive and represents only documents encountered during this research.

Software Development Process

DOD DIR 5000.1	Major System Acquisition
DOD DIR 5000.2	Defense Acquisition Program Procedures
DOD DIR 5000.29	Management of Computer Resources in Major Defense Systems
DOD FAR Sup 70	Acquisition of Computer Resources
DOD-STD-480B	Configuration Control - Engineering Changes, Deviations, And Waivers
DOD-STD-1467	Software Support Environment
DOD-STD-2167A	Defense System Software Development
DOD-STD-2168	Defense System Software Quality Program
DOD HDBK-287	Defense System Software Development Handbook
DOD 5000.3-M-3	Software Test and Evaluation Master Plan Outline
MIL-STD-483A	Configuration Management Practices For Systems, Equipment, Munitions, And Computer Programs
MIL-STD-490A	Specification Practices
MIL-STD-1521B	Technical Reviews and Audits for Systems, Equipments, and Computer Software

Pub Law 89-306	Brooks Bill, Warner-Nunn Amendment and Implementation Directives
AFR 800-14	Life Cycle Management of Computer Resources in Systems
AFSCP 800-5	Software Development Capability/Capacity Review
AFSCP 800-14	Software Quality Indicators
AFSCP 800-43	Software Management Indicators
ESD-TR-88-001	Software Management Metrics

System Safety Program

*	Air Force System Safety Handbook
*	Program Manager's System Safety Guide
MIL-STD-882B	System Safety Program Requirements
MIL-STD 1574A	System Safety Program for Space and Missile Systems
AFSC DH 1-6	Design Handbook: System Safety
AFISC SSH 1-1	Software System Safety Handbook
AFR 80-14	Test and Evaluation
AFR 122-3	The Air Force Nuclear Certification Program
AFR 122-9	Nuclear Surity Design Certification Program For Nuclear Weapon System Software And Firmware
AFR 800-16	USAF System Safety Program
AFSCP 127-1	System Safety Program Management
AFSCP 800-44	System Safety Guidelines

* These unnumbered documents are published by the Air Force Safety Agency

Appendix C: Additional Analysis Techniques/Tools

This appendix lists additional analysis techniques and tools recommended by guidance documents or utilized by various software developers to assist in software safety analysis. None of these techniques and tools, however, is specifically designed to perform structured software safety analysis. The list was compiled from review of references (8:A-19, 18:7-14, 19:21, 32:41, 42:E-21) and a survey of aerospace industry personnel.

- Checklist of Common Software Errors
- Code and Module Walkthroughs
- Compare and Certification Tool
- Critical Function Flows
- Cross Reference Tools
- Data Flow Techniques
- Design and Code Inspections
- Emulation
- Hardware/Software Interface Analysis
- Hierarchy Tool
- Integrated Critical Path Analysis
- Mathematical Proofs
- Proof of Adequacy
- Prototyping
- Reverse Engineering Tools
- Simulation

- Software Control Mapping
- Software Matrices
- Software and Hardware Allocation Analysis
- Software Common Mode Analysis
- Source Code Analysis
- System Cross Check Matrices
- Test Coverage Analysis
- Thread Analysis
- Top-Down Review of Code
- Topology Network Trees
- Traceability Matrices
- Requirements Modeling/Analysis

Appendix D: Software Safety Analysis Survey Questionnaire

PERSON CONTACTED:_____DATE/TIME:_____

PHONE #:_____

COMPANY/DOD AFFILIATION:_____

POSITION/DUTY TITLE:_____

INTERVIEWER:_____

HOW MANY TIMES HAS THIS PERSON BEEN CONTACTED:_____

QUESTIONS:

1. Do you work with the DOD software development process as defined in DOD-STD-2167A or some close derivitive?

2. Is software safety or system safety an issue that is faced in your organization?

3. Is software safety considered internal or external to the overall software development process?

4. Are there any specific software safety analysis techniques and/or management processes related to software safety employed in your organization? Explain.

5. What successes and/or failures has your organization had with software safety analysis techniques?

6. In your opinion, what are the benifits or drawbacks to applying software safety analysis techniques?

REFERALS:

NAME & PHONE:_____

Appendix E: Survey Participants

1. Mr Thomas D. Arkwright, Phd
   System and Software Engineer
   Lockheed Missiles and Space Company
   Palo Alto, CA
   408-756-1861

2. Mr Charles T. Fouts
   Staff Engineer
   Boeing Defense and Space Group Helicopters
   Philadelphia, PA
   215-591-6330

3. Mr Seymour R. Friedman
   Associate Department Head
   The MITRE Corp
   Boston, MA
   617-271-7183

4. Mr Myron D. Krueger
   Engineering Chief
   F-16 Safety Analysis
   General Dynamics, Fort Worth Division
   817-763-3306

5. Mr Charles Lavine
   Software Safety Researcher
   The Aerospace Corporation
   Los Angeles, CA
   310-336-1595

6. Captain Peter Lemieux
   System Safety Manager
   AFSA/SESD
   Norton AFB, CA
   DSN 876-4104

7. Mr Roger Lockwood
   System Safety Manager
   Space and Missile Systems Center
   Los Angeles AFB, CA
   DSN 833-0369

8. Mr Jonathan F. Luedeke
   Principal Research Engineer
   Battelle
   Columbus, OH
   614-424-5145

9. Mr Mitchell F. Lustig
Director, System Safety
Aeronautical Systems Center System Safety Office
Wright-Patterson AFB, OH
DSN 785-3694

10. Captain Steven F. Mattern
System Safety Manager
National Aerospace Plane System Safety Office
Wright-Patterson AFB, OH
DSN 785-1855

11. Mr Archibald McKinlay VI
Software Safety Consultant
McKinlay and Associates
St Louis, MO
314-532-2136

12. Captain Kenneth Miller
Satellite Production Manager
OL-AR, GE Astro-Space Division
Princeton, NJ
609-490-2466

13. Mr James W. Mitchel Jr.
Director, System Safety
Air Force Development Test Center System Safety Office
Eglin AFB, FL
DSN 872-4157

14. Mr Michael C. Moran
Chairman, Electronic Industries Association
G-48 System Safety Committee
Lockheed Corporation
404-494-2083

15. Captain Oscar Overton
Electronic and Software System Safety Manager
USAF/AFOTEC/SEE
Kirtland AFB, NM
DSN 246-5321

16. Mr Douglas A. Peterson
System Safety Engineer
F-22 System Program Office
Wright-Patterson AFB, OH
DSN 785-4502

17. Mr Chris E. Phillips
Safety Engineer
General Dynamics, Fort Worth Division
817-777-5811

18. Mr Patrick Place
Member of the Technical Staff
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

19. Mr William B. Rieger
System Safety Engineer/Software Safety Engineer
Boeing Defense and Space Group
Seattle, WA
206-544-4254

20. Mr Kenneth Roberts
Software Development Engineer
GE Astro-Space Division
Princeton, NJ
609-951-7334

21. Mr W. Coy Sullivan
Chief, E-3 OFP Branch
Oklahoma Air Logistics Center
Tinker AFB, OK
DSN 336-7044

22. Mr Phillip C. Topping
Software Engineer
Lockheed Missiles and Space Company
Sunnyvale, CA
408-742-2780

23. Mr William J. Urschel
Flight Systems Computer Resources
B-2 System Program Office
Wright-Patterson AFB, OH

Appendix F: Hazard Analysis Documents

To maintain a software safety analysis focus, our objective was to review documents produced and delivered under the 300 series tasks of MIL-STD-882B with Notice 1. However, Notice 1 was not introduced until July 1987 and many of the current weapon system acquisition programs were con ıcted before 1987. In addition, one current contract was let before the 1984 publication of MIL-STD-882B and uses MIL-STD-882A. Only a few programs have been able to put the 300 series tasks on contract and fewer still have received the corresponding hazard analysis reports. Therefore, most of the documents reviewed, with one exception, are results of either MIL-STD-882A or the 100 and 200 series tasks of MIL-STD-882B. The one exception is the result of Task 301, Software Requirements Hazard Analysis.

The ASC system program offices that participated in the study and the hazard analysis reports reviewed are listed below. Where applicable each document has a parenthetical reference to its associated MIL-STD-882B, Notice 1 task.

C-17 System Program Office:

- System Safety Program Plan, Rev E (Task 101)
- System Hazard Analysis, Rev B (Task 204)
- Avionics Subsystem Hazard Analysis, Rev B (Task 203)

- Fault Tree Analysis of Erroneous Flight Control Information on the Heads-Up Display
- Electronic Flight Control Sneak Circuit Analysis Final Report, Rev A

These documents are products of the C-17 Full Scale Engineering Development (FSED) phase which was awarded in 1981 when MIL-STD-882A was the current system safety requirements document. The C-17 is currently in its Low Rate Initial Production phase; therefore, no additional software specific hazard analyses have been put on contract.

F-16 System Program Office:

- System Safety Hazard Analysis (Task 204) for the F-16/Full Scale Engineering and Development Low Altitude Navigation and Targeting Infrared for Night (LANTIRN) Navigation Pod System Wide Integrity Management (SWIM) Study

F-22 System Program Office:

- System Safety Program Plan (Task 101)
- Preliminary Hazard Analysis (Task 202)
- A combination of a Subsystem, System, and Operating and Support Hazard Analysis (Tasks 203, 204, and 205)
- Safety Assessment Report (Task 209)

These documents were produced for the F-22 Demonstration/Validation phase and were used by the program office to aid in the F-22 aircraft's proof of concept. The F-22 program has recently entered the Engineering and Manufacturing Development phase and MIL-STD-882B 300 series tasks have been imposed on the contractor developing team.

Electronic Combat and Reconnaissance (RW) System

Program Office:

- Software Requirement Hazard Analysis (Task 301), for the EF-111A, System Improvement Program (SIP), AN/ALQ-99 Encoder/Processor.

Appendix G: System Safety Managers Interviewed

1. Mr Eldon Bertran
   Modern Technologies Corporation
   System Safety Management Contractor
   F-22 System Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-4502

2. Major Robert J. Congelli
   Director, System Safety
   C-17 System Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-6719

3. Lt Col Harry V. Dutchyshyn
   Chief, F-16 System Safety Division
   F-16 System Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-1938

4. Mr Jon D. Kocara
   Director, System Safety
   Electronic Combat & Reconnaissance Sys Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-9249

5. Captain Steven F. Mattern
   Chief, System Safety Management
   NASP System Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-1855

6. Mr Douglas A. Peterson
   System Safety Engineer
   F-22 System Program Office
   Wright-Patterson AFB, OH 45433
   DSN 785-4502

Appendix H: Points of Contact For System Safety Training

ASC Training Course

For more information on the ASC System Safety Training Course contact:

Mr Mitchell F. Lustig
Director, System Safety
ASC/EMSS
Wright-Patterson AFB, OH 45433
DSN 785-3694

Air Force Sponsored Courses

For complete course descriptions, prerequisites, course lengths, and locations of Air Force sponsored courses refer to the latest version of AFR 50-5, Training: USAF Formal Schools.

For information on SAS 001, 002, and 006 courses contact:

6575th School Squadron (AFMC)
Brooks AFB, TX 78235-5000
DSN 240-2102

Navy Sponsored Courses

For more information on the system safety courses conducted by the Naval Safety School contact:

Naval Safety School
Naval Air Station
Norfolk, VA 23511-5410
DSN 565-8778

Commercially Available Courses

For more information on the System Safety Course and the Software Safety Course contact:

University of Southern California
Institute of Safety and Systems Management
927 West 35th Place
Room 102
Los Angeles, CA 90089-0021
Phone: 213-740-3995

For more information on the Software Safety Workshop contact:

University of California-Los Angeles
Extension Building
Short Course Program Office
10995 Le Conte Avenue
Suite 542
Los Angeles, CA 90024-2883
Phone: 310-825-3344

Bibliography

1. Brown, Michael L. "Software Systems Safety And Human Errors," IEEE Compass. 19-28. 1988.

2. -----. Software Systems Safety Design Guidelines and Recommendations. NSWC-TR-89-33. Dahlgren VA: Naval Surface Warfare Center, March 1989 (AD-A209832).

3. Buratti, Davey L. and Sylvia G. Godoy. Sneak Analysis Application Guidelines. RADC-TR-82-179. Houston TX: Boeing Aerospace Company, June 1982 (AD-A118479).

4. Cha, Stephen S., Nancy G. Leveson, and Timothy J. Shimeall. "Safety Verification in Murphy Using Fault Tree Analysis," Proceedings of 10th International Conference on Software Engineering. 377-386. Singapore, 1988.

5. Defense Systems Management College. Mission Critical Computer Resources Management Guide. Washington: Government Printing Office, 1990.

6. Department of Defense. Configuration Control-Engineering Changes, Deviations and Waivers. DOD-STD-480B. Washington: Government Printing Office, 15 July 1988.

7. -----. Defense System Software Development. DOD-STD-2167A. Washington: Government Printing Office, 29 February 1988.

8. -----. System Safety Program Requirements. MIL-STD-882B with Notice 1. Washington: Government Printing Office, July 1987.

9. -----. Technical Reviews And Audits For Systems, Equipments, And Computer Software. MIL-STD-1521B. Washington: Government Printing Office, 4 June 1985.

10. Department of the Air Force. Program Manager's System Safety Guide. Norton AFB CA: Air Force Safety Agency, March 1992.

11. -----. Safety: System Safety Management. ASD Pamphlet 127-1. Wright-Patterson AFB OH: HQ ASD (AFSC), 11 March 1985.

12. -----. Software Management Guide. Hill AFB UT: Software Technology Support Center, April 1992.

13. -----. System Safety Handbook: Software System Safety. AFISC SSH 1-1. Norton AFB CA: Air Force Inspection and Safety Center, 5 September 1985.

14. -----. Training: USAF Formal Schools. AFR 50-5. Washington: HQ USAF, 1 September 1991.

15. Emory, C. William and Donald R. Cooper. Business Research Methods (Fourth Edition). Homewood IL: Richard D. Irwin, Inc., 1991.

16. Fagan, M. E. "Design And Code Inspections To Reduce Errors In Program Development," IBM Systems Journal, 15-3: 219-248. 1976.

17. Firth, Robert and others. A Guide to the Classification and Assessment of Software Engineering Tools. Pittsburgh: Software Engineering Institute, 1987.

18. Frola, Ronald F. and C. O. Miller. System Safety In Aircraft Acquisition. Contract MDA903-81-C-0166 (Task ML214). Washington DC: Logistics Management Institute, January 1984.

19. Forrest, Maurice. "Software Safety," Hazard Prevention, 24: 20-21 (July/September 1988)

20. Gellman, Barton. "Computer Problem Cited in Crash of F-22 Prototype," Washington Post, 115: A3 (30 April 1992).

21. Hayward, Duston L. A Practical Application Of Petri Nets In The Software Safety Analysis Of A Real-Time Military System. MS Thesis. Naval Postgraduate School, Monterey CA, December 1987 (AD-A188993).

22. Hughes, David. "Computer Experts Discuss Merits Of Defense Dept. Software Plan," Aviation Week & Space Technology, 132: 65 (16 April 1990).

23. Jahanian, Farnam and Aloysius K. Mok. "Safety Analysis of Timing Properties in Real-Time Systems," IEEE Transactions on Software Engineering, SE-12: 890-902 (September 1986).

24. Leveson, Nancy G. "Software Safety: Why, What, and How," Computing Surveys, 18: 125-163 (June 1986).

25. -----. "Evaluation of Software Safety," Proceedings of 12th International Conference on Software Engineering. 223-224. Nice, France, 1990.

26. -----. "Software Safety in Embedded Computer Systems," Communications of the ACM, 34: 35-46 (February 1991).

27. -----, Stephen S. Cha, and Timothy J. Shimeall. "Safety Verification of Ada Programs Using Software Fault Trees," IEEE Software, 48-59 (July 1991).

28. -----. and Janice L. Stolzy. "Safety Analysis Using Petri Nets," IEEE Transactions on Software Engineering, SE-13: 386-397 (March 1987).

29. Lustig, Mitchell F. Course viewgraphs for System Safety Manager's in-house introductory training. Directorate of Safety, Aeronautical Systems Center (AFMC), Wright-Patterson AFB OH, June 1991.

30. -----, Director, System Safety. Personal interview. Aeronautical Systems Center (AFMC), Wright-Patterson AFB OH, 22 June 1991.

31. Madhavji, Nazim H. "The Process Cycle," Software Engineering Journal, 234-241 (September 1991).

32. Mattern, Steven F. Software System Safety. MA Thesis. Department of Computer Resource Management, Webster University, St. Louis MO, December 1988

33. Neumann, Peter G. "On Hierarchical Design of Computer Systems for Critical Applications," IEEE Transactions on Software Engineering, SE-12: 905-920 (September 1986).

34. -----. "Risks To The Public In Computer Systems: Critical Systems," ACM SIGSOFT Software Engineering Notes, 9: 2 (October 1984).

35. -----. "Risks To The Public In Computer Systems: F-16 Problems," ACM SIGSOFT Software Engineering Notes, 11: 6 (October 1986).

36. -----. "Risks To The Public In Computer Systems: More Disaster Reports," ACM SIGSOFT Software Engineering Notes, 8: 3 (October 1983).

37. -----. "Some Computer-Related Disasters And Other Egregious Horrors," ACM SIGSOFT Software Engineering Notes, 10: 6-7 (January 1985).

38. -----. "The N Best (or Worst) Computer-Related Risk Cases," IEEE Compass. xi-xiii. 1987.

39. Parnas, David L., A. John van Schouwen, and Shu Po Kwan. "Evaluation of Safety-Critical Software," Communications of the ACM, 33: 636-648 (June 1990).

40. Pressman, Roger S. Software Engineering: A Practitioner's Approach (Second Edition). New York: McGraw-Hill Publishing Company, 1987.

41. Stewart, Nick. "Software Error Costs," Quality Progress, 21: 48-49 (November 1988).

42. System Safety in Software Development. Education Program Course 2X-94206. System Safety Department, Boeing Aerospace and Electronics, Seattle WA, September 1991.

43. Tapscott, Mark. "Washington Report: F-14D Embedded Software Not Reliable, GAO Contends," Defense Electronics, 24: 10 (July 1992).

44. Wray, Tony. "The Everyday Risks of Playing Safe," New Scientist, 1629: 61-65 (September 1988).

Vita

Captain Peter W. Colan was born on 30 June 1957 in Heidelberg, Germany. He graduated from Karlsruhe American High School in Karlsruhe, Germany in 1976. He enlisted in the United States Air Force in April 1978 and served for over six years as an Avionic Sensor System Specialist working on SR-71 aircraft at Beale Air Force Base, California and Kadena Air Base, Japan. He was selected for the Airman Education and Commissioning Program in August 1984 and attended Wright State University in Dayton, Ohio where he earned a Bachelor of Science degree in Electrical Engineering in 1987. Upon graduation, he attended Officer Training School and accepted a commission in the United States Air Force as a Second Lieutenant. He was subsequently assigned to Los Angeles Air Force Base, California where he served as a System Acquisition Manager in the Defense Meteorological Satellite Program's Future Satellite Systems Division. In May 1991, he entered the School of Systems and Logistics, Air Force Institute of Technology.

Permanent Address: 104 Cottonwood Court
Sterling, VA 20164

Vita

Captain Robert W. Prouhet was born on 6 February 1958 in St. Charles, Missouri. He graduated from Berkeley Senior High School in Berkeley, Missouri in 1976. He enlisted in the USAF in November 1977 and entered active duty in December 1977 as an Avionic Instrument Systems Specialist. After serving five years at Dover AFB, Delaware, he was selected for the Airman Education and Commissioning Program. In December 1985 he graduated from the University of Missouri in Columbia, Missouri, with a Bachelor of Science in Electrical Engineering and immediately entered Officer Training School (OTS). After graduating from OTS in April 1986, he served as an Aircraft Electronic Support Equipment Engineer in the B-1B and F-16 System Program Offices at Wright-Patterson AFB, Ohio. There he performed a broad spectrum of engineering and management tasks in the acquisition of depot and flightline support equipment for the B-1B and F-16 aircraft. In May 1991 he entered the School of Systems and Logistics, Air Force Institute of Technology.

Permanent Address: 723 N. Yosemite Court
St Peters, MO 63376

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
AN ASSESSMENT OF SOFTWARE SAFETY AS APPLIED TO THE DEPARTMENT OF DEFENSE SOFTWARE DEVELOPMENT PROCESS

5. FUNDING NUMBERS

6. AUTHOR(S)
Peter W. Colan, Captain, USAF
Robert W. Prouhet, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GSS/ENG/92D-2

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This research analyzed the relationships between the DOD software development process, system safety requirements, and current structured software safety analysis techniques. The current state of software safety was assessed within the aerospace industry and DOD, and a training program for DOD System Safety Mana rs was developed. A telephone survey was conducted to gather information urrent software safety analysis techniques and methodologies. Personal interviews were conducted with Aeronautical System Center System Safety Managers to gather data on job perception and perceived training needs. The results of the study indicate that the DOD guidance and policy documents needed to implement and manage a software safety program are adequate. Software safety programs, however, are not implemented and managed effectively; in addition, structured software safety analysis techniques are not widely used. To improve the DOD's ability to manage weapon system safety programs, DOD should implement the proposed training program.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Safety Program Management, Safety Training, Safety Engineering, System Safety, Software Development, Software Safety Analysis	139
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

AFIT Control Number AFIT/GSS/ENG/92D-2

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: AFIT/LSC, Wright-Patterson AFB OH 45433-9905.

1. Did this research contribute to a current research project?

a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____

Name and Grade Organization

_____ _____

Position or Title Address